Discrete Two-Stage Stochastic Mixed-Integer Programs with Applications to Airline Fleet Assignment and Workforce Planning Problems

Xiaomei Zhu

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Dr. Hanif D. Sherali, Chair
Dr. Ebru K. Bish
Dr. Kyle Y. Lin
Dr. Subhash C. Sarin
Dr. Antonio A. Trani

April, 2006
Blacksburg, Virginia

Keywords: Stochastic Programming, Mixed-Integer Programming, Stochastic Mixed-Integer Programming, Reformulation-Linearization Technique (RLT), Benders' Decomposition, Airline Fleet Assignment, Workforce Planning Problem

Discrete Two-Stage Stochastic Mixed-Integer Programs with Applications to Airline Fleet Assignment and Workforce Planning Problems

Xiaomei Zhu

(Extended Abstract)

Stochastic programming is an optimization technique that incorporates random variables as parameters. Because it better reflects the uncertain real world than its traditional deterministic counterpart, stochastic programming has drawn increasingly more attention among decision-makers, and its applications span many fields including financial engineering, health care, communication systems, and supply chain management. On the flip side, stochastic programs are usually very difficult to solve, which is further compounded by the fact that in many of the aforementioned applications, we also have discrete decisions, thereby rendering these problems even more challenging. In this dissertation, we study the class of *two-stage stochastic mixed-integer programs* (SMIP), which, as its name suggests, lies at the confluence of two formidable classes of problems. We design a novel algorithm for this class of problems, and also explore specialized approaches for two related real-world applications.

Although a number of algorithms have been developed to solve two-stage SMIPs, most of them deal with problems containing *purely* integer or continuous variables in either or both of the two stages, and frequently require the technology and/or recourse matrices to be deterministic. As a ground-breaking effort, in this work, we address the challenging class of two-stage SMIPs that involve 0-1 mixed-integer variables in both stages. The only earlier work on solving such problems (Carøe and Schultz (1999)) requires the optimization of several non-smooth Lagrangian dual problems using subgradient methods in the bounding process, which turns out to be computationally very expensive.

We begin with proposing a decomposition-based branch-and-bound (**DBAB**) algorithm for solving two-stage stochastic programs having 0-1 mixed-integer variables in both stages. Since the second-stage problems contain binary variables, their value functions are in general nonconvex and discontinuous; hence, the classical Benders' decomposition approach (or the L-shaped method) for solving two-stage stochastic programs, which requires convex subproblem value functions, cannot be directly applied. This motivates us to relax the second-stage problems and accompany this relaxation with a convexification process. To make this process computationally efficient, we propose to construct a certain partial convex hull representation of the two-stage solution space, using the relaxed second-stage constraints and the restrictions confining the first-stage variables to lie within some hyperrectangle. This partial convex hull is sequentially generated using a convexification scheme, such as the Reformulation-Linearization Technique (RLT), which yields valid inequalities that are functions of the first-stage variables and, of noteworthy importance, are reusable in the subsequent subproblems by updating the values of the first-stage variables. Meanwhile, since the first stage contains continuous variables, whenever we tentatively fix these variables at some given feasible values, the resulting constraints may not be facial with respect

to the associated bounding constraints that are used to construct the partial convex hull. As a result, the constructed Benders' subproblems define *lower bounds* for the second-stage value functions, and likewise, the resulting Benders' master problem provides a lower bound for the original stochastic program defined over the same hyperrectangle. Another difficulty resulting from continuous first-stage variables is that when the given first-stage solution is not extremal with respect to its bounds, the second-stage solution obtained for a Benders' subproblem defined with respect to a partial convex hull representation in the two-stage space may not satisfy the model's binary restrictions. We thus need to be able to detect whether or not a Benders' subproblem is solved by a given fractional second-stage solution. We design a novel procedure to check this situation in the overall algorithmic scheme. A key property established, which ensures global convergence, is that these lower bounds become exact if the given first-stage solution is a vertex of the defining hyperrectangle, or if the second-stage solution satisfies the binary restrictions.

Based on these algorithmic constructs, we design a branch-and-bound procedure where the branching process performs a hyperrectangular partitioning of the projected space of the first-stage variables, and lower bounds for the nodal problems are computed by applying the proposed modified Benders' decomposition method. We prove that, when using the least-lower-bound node-selection rule, this algorithm converges to a global optimal solution. We also show that the derived RLT cuts are not only reusable in subsequent Benders iterations at the same node, but are also inheritable by the subproblems of the children nodes. Likewise, the Benders' cuts derived for a given sub-hyperrectangle can also be inherited by the lower bounding master programs solved for its children nodes. Using these cut inheritance properties results in significant savings in the overall computational effort.

Some numerical examples and computational results are presented to demonstrate the efficacy of this approach. The sizes of the deterministic equivalent of our test problems range from having 386 continuous variables, 386 binary variables, and 386 constraints, up to 1795 continuous variables, 1539 binary variables, and 1028 constraints. The results reveal an average savings in computational effort by a factor of 9.5 in comparison with using a commercial mixed-integer programming package (CPLEX 8.1) on a deterministic equivalent formulation.

We then explore an important application of SMIP to enhance the traditional airline fleet assignment models (FAM). Given a flight schedule network, the fleet assignment problem solved by airline companies is concerned with assigning aircraft to flight legs in order to maximize profit with respect to captured path- or itinerary-based demand. Because certain related crew scheduling regulations require early information regarding the type of aircraft serving each flight leg, the current practice adopted by airlines is to solve the fleet assignment problem using estimated demand data 10-12 weeks in advance of departure. Given the level of uncertainty, deterministic models at this early stage are inadequate to obtain a good match of aircraft capacity with passenger demands, and revisions to the initial fleet assignment become naturally pertinent when the observed demand differs considerably from the assigned aircraft capacities. From this viewpoint, the initial decision should embrace

various market scenarios so that it incorporates a sufficient look-ahead feature and provides sufficient flexibility for the subsequent re-fleeting processes to accommodate the inevitable demand fluctuations.

With this motivation, we propose a two-stage stochastic programming approach in which the first stage is concerned with the initial fleet assignment decisions and, unlike the traditional deterministic methodology, focuses on making only a family-level assignment to each flight leg. The second stage subsequently performs the detailed assignments of fleet types within the allotted family to each leg under each of the multiple potential scenarios that address corresponding path- or itinerary-based demands. In this fashion, the initial decision of what aircraft family should serve each flight leg accomplishes the purpose of facilitating the necessary crew scheduling decisions, while judiciously examining the outcome of future re-fleeting actions based on different possible demand scenarios. Hence, when the actual re-fleeting process is enacted several weeks later, this anticipatory initial family-level assignment will hopefully provide an improved overall fleet type re-allocation that better matches demand. This two-stage stochastic model is complemented with a secondary model that performs adjustments within each family, if necessary, to provide a consistent fleet type-assignment information for accompanying decision processes, such as yield management. We also propose several enhanced fleet assignment models, including a robust optimization model that controls decision variation among scenarios and a stochastic programming model that considers the recapture effect of spilled demand.

In addition to the above modeling concepts and framework, we also contribute in developing effective solution approaches for the proposed model, which is a large-scale two-stage stochastic 0-1 mixed-integer program. Because the most pertinent information needed from the initial fleet assignment is at the family level, and the type-level assignment is subject to change at the re-fleeting stage according to future demand realizations, our solution approach focuses on assigning aircraft families to the different legs in the flight network at the first stage, while finding relaxed second-stage solutions under different demand scenarios. Based on a polyhedral study of a subsystem extracted from the original model, we derive certain higher-dimensional convex hull as well as partial convex hull representations for this subsystem. Accordingly, we propose two variants for the primary model, both of which relax the binary restrictions on the second-stage variables, but where the second variant then also accommodates the partial convex hull representations, yielding a tighter, albeit larger, relaxation. For each variant, we design a suitable solution approach predicated on Benders' decomposition methodology. Using certain realistic large-scale flight network test problems having 900 flight legs and $1,814$ paths, as obtained from *United Airlines*, the proposed stochastic modeling approach was demonstrated to increase daily expected profits by about 3% (which translates to about $160 million per year) in comparison with the traditional deterministic model in present usage, which considers only the expected demand. Only 1.6% of the second-stage binary variables turn out to be fractional in the first variant, and this number is further reduced to 1.2% by using the tighter variant. Furthermore, when attempting to solve the deterministic equivalent formulation for these two variants using a

commercial mixed-integer programming package (CPLEX 8.1), both the corresponding runs were terminated after reaching a 25-hour cpu time limit. At termination, the software was still processing the initial LP relaxation at the root node for each of these runs, and no feasible basis was found. Using the proposed algorithms, on the other hand, the solution times were significantly reduced to 5 and 19 hours for the two variants, respectively. Considering that the fleet assignment models are solved around three months in advance of departure, this solution time is well acceptable at this early planning stage, and the improved quality in the solution produced by considering the stochasticity in the system is indeed highly desirable.

Finally, we address another practical workforce planning problem encountered by a global financial firm that seeks to manage multi-category workforce for functional areas located at different service centers, each having office-space and recruitment-capacity constraints. The workforce demand fluctuates over time due to market uncertainty and dynamic project requirements. To hedge against the demand fluctuations and the inherent uncertainty, we propose a two-stage stochastic programming model where the first stage makes personnel recruiting and allocation decisions, while the second stage, based on the given personnel decision and realized workforce demand, decides on the project implementation assignment. The second stage of the proposed model contains binary variables that are used to compute and also limit the number of changes to the original plan. Since these variables are concerned with only one quality aspect of the resulting workforce plan and do not affect feasibility issues, we replace these binary variables with certain conservative policies regarding workforce assignment change restrictions in order to obtain more manageable subproblems that contain purely continuous variables. Numerical experiments reveal that the stochastic programming approach results in significantly fewer alterations to the original workforce plan. When using a commercial linear programming package CPLEX 9.0 to solve the deterministic equivalent form directly, except for a few small-sized problems, this software failed to produce solutions due to memory limitations, while the proposed Benders' decomposition-based solution approach consistently solved all the practical-sized test problems with reasonable effort.

To summarize, this dissertation provides a significant advancement in the algorithmic development for solving two-stage stochastic mixed-integer programs having 0-1 mixed-integer variables in both stages, as well as in its application to two important contemporary real-world applications. The framework for the proposed solution approaches is to formulate tighter relaxations via partial convex hull representations and to exploit the resulting structure using suitable decomposition methods. As decision robustness is becoming increasingly relevant from an economic viewpoint, and as computer technological advances provide decision-makers the ability to explore a wide variety of scenarios, we hope that the proposed algorithms will have a notable positive impact on solving stochastic mixed-integer programs. In particular, the proposed stochastic programming airline fleet assignment and the workforce planning approaches studied herein are well-poised to enhance the profitability and robustness of decisions made in the related industries, and we hope that similar improve-

ments are adapted by more industries where decisions need to be made in the light of data that is shrouded by uncertainty.

This work is based on research supported by the *National Science Foundation* under Grant Numbers DMI-0094462 and DMI-0245643.

Dedicated to my parents, sister, and the memory of my grandfather
献给我的父母, 妹妹, 和爷爷

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. Hanif D. Sherali, for his guidance, support, and patience throughout my Master and Ph.D. study. He is truly a master in advising students, conducting research, and teaching, and more than any thing, he is a role model for being so humble even when having achieved so much. It is indeed a privilege to work with him. I owe much to my committee member and Master's thesis co-advisor Dr. Ebru K. Bish, who has always been a great mentor and friend, and who has helped me in many ways, including and beyond my dissertation. I am grateful to my committee members Dr. Subhash C. Sarin, Dr. Antonio A. Trani, and Dr. Kyle Y. Lin for their valuable help and guidance in my dissertation, coursework, and academic career pursuit. I also appreciate Dr. Kimberly P. Ellis for providing me access to the Dover (MSI) lab.

My heartfelt thanks to my friends in Blacksburg for making my life here so wonderful. I thank Morr (Pornthipa Ongkunaruk) and Jennifer (Peifang Tsai) for being the best roommates I could ever ask for. I thank Weiping Chen for all his technical, equipment, and emotional support over the past two years. I also thank Feng Li for his encouragement all the time—whenever I need it. And many, many others, Xiaopei Chen, Nipun Palekar, Milind Mohile, Li Cai, Lian Jian, Lixin Wang, Yuqiang Wang, ... thank you all for the good times we shared.

Finally, overwhelmingly thanks to my parents and sister for their unconditional love and constant support. I thank my father (Jihua Zhu) for teaching me math and English when I was little; I thank my mother (Menghua Chen) for constantly trying to inculcates optimistic attitude in me; and I thank my sister (Xiaodan Zhu) for her absolute faith in me. I appreciate their understanding and tolerance in accepting my absence from home all these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Stochastic programs (SP) have drawn increasingly more attention in the optimization arena because they incorporate random variables as parameters, which more closely reflects the uncertain real world. Meanwhile, mixed-integer programs (MIP) have been studied extensively and have become well-established decision tools in many industries. In the past decade, algorithmic developments in both areas, along with advancements in computer technology, have induced the study of problems that incorporate both these modeling paradigms—the class of stochastic mixed-integer programs (SMIP). This dissertation is concerned with the study of theory and algorithms, along with implementation aspects and applications of stochastic mixed-integer programming problems.

1.1 Motivation

Arguably, SMIPs are among the most challenging of optimization problems because both SPs and MIPs are generally difficult to solve. A special case of SMIPs that has been actively studied is the class of two-stage stochastic mixed-integer programming problems with recourse. However, most of the solution approaches proposed for this class of problems have thus far assumed purely continuous or purely binary variables in either or both stages. The challenges posed by the design of algorithms for general two-stage SMIPs having mixed-integer variables in both stages are manifold. First of all, a common difficulty of solving SPs is that the incorporation of randomness typically yields very large sized problems. The problem may contain many random parameters, each having a large number of potential realizations. The size of the SP can easily explode as the number of random parameters and their realizations increase. Hence, solution approaches for SMIPs must contend with the development of algorithms for large-scale MIPs, which is already a challenging task by itself. A well-studied method for solving large-scale programs, including MIPs, is the Benders' decomposition method (Benders (1962)), also called the "L-shaped method" in the SP arena (Van

Slyke and Wets (1969), Laporte and Louveaux (1993)). When applying Benders' decomposition to two-stage stochastic programs, the master problem includes the first-stage variables and the subproblems are the second-stage problems. The application of this methodology to general stochastic mixed-integer programs brings along several significant obstacles: (a) non-convex and discontinuous second-stage value functions when the subproblem contains integer variables, and (b) the potentially infinite branching on the first-stage variables with a possibility of non-convergence when the master program includes continuous first-stage variables.

Sherali and Fraticelli (2002) have modified the basic Benders' decomposition framework to render it suitable for solving two-stage stochastic programs having purely binary first-stage variables and mixed-integer second-stage variables. In their method, the (partial) convex hull of the second-stage feasible solutions is constructed using a convexification process—the Reformulation-Linearization Technique (RLT) process (see Sherali and Adams (1990, 1994, 1999)). The generated convexification cuts are functions of the first-stage *variables*, instead of simply pertaining to given first-stage solution *values* obtained for each specific iteration, so that the cuts can be used repeatedly each time the first-stage solution values are changed. The theoretical convergence of this algorithm, however, assumes purely binary first-stage variables. When the first stage contains continuous variables, the different feasible first-stage solutions generated are not necessarily facial with respect to the defining (bounding) constraints, and the convexification process for the second-stage subproblem is then required to be conducted with respect to nonextremal first-stage variable values. As a result, even if the entire convex hull of the second-stage solution space is constructed, the second-stage solution is not guaranteed to satisfy the binary restrictions, given such nonextremal facial first-stage solutions.

A principal motivation of our research is to design a suitable decomposition algorithm for solving the general class of SMIPs that contain 0-1 mixed-integer variables in both stages. To deal with the nonextremal first-stage variable values in the second-stage convexification process, we apply a branch-and-bound scheme based on a hyperrectangular partitioning process in the projected space of the first-stage variables. We show that the Benders' master problem defined over the hyperrectangle provides a lower bound for the original stochastic program over that region, and furthermore, if the first-stage variable solution for the lower bounding master problem turns out to be an extreme point of the defining hyperrectangle, the objective value obtained from this lower bounding master problem is the same as that for the original stochastic program over the defined region. As mentioned earlier, a side-effect of the continuum of first-stage variable values is that the lower bounding second-stage functional evaluations may not yield binary second-stage solutions even when the subproblems are solved to optimality. To resolve this issue, we design a precise mechanism to detect when to stop solving the second-stage subproblems. Although the first-stage variable domain can potentially be partitioned infinitely, we prove that this algorithm converges to a global optimal solution. We also provide insights into its inter-connection with disjunctive programming-based methods, along with implementation expedients governing the potential

reuse of generated convexification cuts across scenarios. We present some numerical examples and computational results to demonstrate the efficacy of the proposed approach. The sizes of the deterministic equivalent of our test problems range from having 386 continuous variables, 386 binary variables, and 386 constraints, up to 1795 continuous variables, 1539 binary variables, and 1028 constraints. The results reveal an average savings in computational effort by a factor of 9.5 in comparison with using a commercial mixed-integer programming package (CPLEX 8.1) on a deterministic equivalent formulation.

In addition to the foregoing development for general stochastic mixed-integer programs, we study two particular important applications of this class of problems. The first of these is an airline fleet assignment problem that considers uncertainties in demand. The current airline fleet assignment practice is to assign aircraft capacity to scheduled flights well in advance of departures due to crew-related restrictions. We propose a stochastic mixed-integer programming model to advantageously *delay* (or postpone) irrevocable fleeting decisions and to inject sufficient *flexibility* in the initial assignment decisions so as to facilitate the ability to make future revisions within the operational constraints. Even with deterministic demand, the fleet assignment problem is already a very large-scale optimization problem. Considering uncertainties in demand, we need to take extra care in developing solution approaches that can be implemented in practice for real-life airline operations. For this purpose, we conduct a polyhedral study and, accordingly, design suitable solution approaches for two relaxed variants of the primary model based on Benders' decomposition methodology. Using certain realistic large-scale flight network test problems having up to 900 flight legs and $1,814$ paths, as obtained from *United Airlines*, the proposed stochastic modeling approach was demonstrated to increase daily expected profits by about 3% (which translates to about $160 million per year) in comparison with the traditional deterministic model in present usage, which considers only the expected demand. Furthermore, when attempting to solve the deterministic equivalent formulation for the two proposed variants using a commercial mixed-integer programming package (CPLEX 8.1), both the corresponding runs were terminated after reaching a 25-hour cpu time limit. At termination, the software was still processing the initial LP relaxation at the root node for each of these runs, and no feasible basis was found. Using the proposed algorithms, on the other hand, the solution times were significantly reduced to 5 and 19 hours for the two variants, respectively. Considering that the fleet assignment models are solved around three months in advance of departure, this solution time is well acceptable at this early planning stage, and the improved quality in the solution produced by considering the stochasticity in the system is indeed highly desirable.

The second application of stochastic mixed-integer programming that we examine concerns a multi-category workforce planning problem under workload fluctuations and uncertainty, and subject to facility and recruitment capacity restrictions. This application is motivated by a real-life workforce planning problem faced by a global financial firm, and is induced by its outsourcing practices. We formulate this problem as a two-stage stochastic mixed-integer programming model where the personnel related decisions, which take longer to implement due to recruitment and training requirements, are made in the first stage, and

the relatively easier workload distribution decisions are postponed to the second stage. The second stage contains binary variables for assessing, and also limiting, the number of changes to the prescribed plan. We design a solution approach that replaces these binary variables with certain conservative policies regarding workforce assignment change restrictions in order to obtain more manageable subproblems that contain purely continuous variables. We show that compared with a deterministic model that considers only expected workforce demand, the stochastic programming approach results in significantly fewer alterations to the prescribed workforce plan for nontrivial cases, and the proposed Benders' decomposition-based solution approach is reliable in solving practical-sized test problems.

1.2 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 provides a review of the literature relevant to stochastic programming, the fleet assignment problem, and the workforce planning problem. Chapter 3 proposes a decomposition-based branch-and-bound (**DBAB**) algorithm for solving two-stage stochastic programs having mixed-integer first- and second-stage variables. Chapters 4 and 5 present the two aforementioned applications of SMIP, related respectively to the airline fleet assignment problem and the multi-category workforce planning problem. Chapter 6 summarizes our research contributions and suggests future research directions.

Chapter 2

Literature Review

This chapter provides a literature review for the three topics covered in this research and closely related areas. Section 2.1 reviews solution approaches for stochastic linear programs and surveys in detail some recent algorithmic developments in solving stochastic mixed-integer programs. Section 2.2 discusses airline fleet assignment models and re-fleeting models, as well as existing solution approaches. Section 2.3 briefly discusses some workforce planning methods, including both descriptive and normative models.

2.1 Two-Stage Stochastic Programs with Recourse

The current stochastic programming (SP) arena includes two primary types of models: *the stochastic program with recourse models* and *the stochastic program with chance/probability-constraints*. In the recourse models, a set of decisions have to be made *a priori* in a context when the related environmental information is not completely available. These decisions are usually called the *first-stage decisions*. Given the first-stage decisions, the later stage decision variables (also called *recourse variables*) can be decided based on the realization of a number of random events. These recourse variables are also interpreted as "correction actions" if they are used to compensate any infeasibility from the first-stage decisions. Stochastic programs with chance-constraints, on the other hand, allow occasional infeasibility and require that the constraints be satisfied with some specified probability.

In this research, we focus on the two-stage recourse models. These models were first proposed independently by Dantzig (1955) and Beale (1955). Their properties were extensively studied in the 60s and 70s (see, for example, Wets (1966a,b) for SPs having only random right-hand-side (RHS), and Rockafellar and Wets (1974) and Walkup and Wets (1967) for the general cases), and different solution approaches along with variations and extensions have been widely studied over the past two decades. Naturally, compared with stochastic programs having purely continuous variables, those containing integer variables pose an ex-

tra challenge due to their discrete nature. Researchers have actively studied the properties of and solution approaches for these problems. We refer the reader to the extensive survey papers by Schultz et al. (1996), Klein Haneveld and van der Vlerk (1999), Schultz (2003), and Sen (2005), and an annotated bibliography by Stougie and van der Vlerk (1997) for a discussion on the principal properties of SMIPs and some earlier algorithmic developments in this area. In this section, after a brief review on some solution approaches for stochastic linear (continuous) programs, we will focus on certain recent algorithmic advances in solving two-stage stochastic (mixed-) integer programs. Many of these methods are theoretically extendable to multi-stage SMIPs. However, scalability becomes a major issue here, which requires further research. For an introduction on multi-stage stochastic integer programs, we refer the reader to Römisch and Schultz Römisch and Schultz (2001).

A two-stage stochastic (mixed-integer) program with recourse can be formulated as follows:

$$\begin{aligned}
\textbf{S(MI)P:} \quad & \text{Minimize} \;\; cx + E[f(x,\tilde{\omega})] && \text{(2.1a)}\\
\text{subject to} \quad & Ax \geq b && \text{(2.1b)}\\
& x \geq 0 && \text{(2.1c)}\\
& x_i \text{ binary}, \;\; \forall i \in I_b \subseteq I \equiv \{1, ..., n_1\} && \text{(2.1d)}\\
& x_i \text{ integer}, \;\; \forall i \in I_{int} \subseteq I \setminus I_b, && \text{(2.1e)}
\end{aligned}$$

where $\tilde{\omega}$ is a random variable defined on a probability space $(\tilde{\Omega}, \tilde{\mathcal{A}}, \tilde{\mathcal{P}})$ (with $\tilde{\Omega}$, $\tilde{\mathcal{A}}$, and $\tilde{\mathcal{P}}$ respectively denoting the set of all outcomes, a collection of random variables, and their associated probability distributions), and where for any given realization ω of $\tilde{\omega}$, we have the second-stage problem (also called *scenario subproblem*, or simply *subproblem*):

$$\begin{aligned}
f(x,\omega) = \quad & \text{minimum} \quad g(\omega)y && \text{(2.1f)}\\
& \text{subject to} \quad W(\omega)y \geq r(\omega) - T(\omega)x && \text{(2.1g)}\\
& y \geq 0 && \text{(2.1h)}\\
& y_j \text{ binary}, \;\; \forall j \in J_b \subseteq J \equiv \{1, ..., n_2\} && \text{(2.1i)}\\
& y_j \text{ integer}, \;\; \forall j \in J_{int} \subseteq J \setminus J_b. && \text{(2.1j)}
\end{aligned}$$

In the above, A is an $m_1 \times n_1$ matrix, and for each ω, $W(\omega)$ is an $m_2 \times n_2$ *recourse matrix*, $T(\omega)$ is an $m_2 \times n_1$ *technology matrix*, and the other defined vectors are of corresponding conformable sizes. We assume that the elements of A, $T(\omega)$, and $W(\omega)$ are rational (so that they are scalable to integers, if necessary). It is often assumed that the second-stage problem is feasible for any given first-stage decision x, a property called *complete recourse*. Usually, it is sufficient to have a weaker condition, *relatively complete recourse*, which requires that the second-stage problem is feasible for any *feasible* first-stage decision x. For computational viability, a finite number of scenarios, denoted as a set S and indexed by s, is often considered based on some discretization of the possible realizations of $\tilde{\omega}$, each with an associated

probability of occurrence p_s, $s \in S$. (See Schultz (1995) for a justification on approximating continuously distributed scenario parameters by a discrete distribution having a finite support.) Accordingly, the realizations of $g(\omega)$, $W(\omega)$, $T(\omega)$, and $r(\omega)$ are correspondingly denoted as g_s, W_s, T_s, and r_s, respectively, for $s \in S$. The second-stage *value function* $f(\cdot, \omega)$ is also accordingly denoted as $f_s(\cdot)$, for $s \in S$.

Collecting the problems for the two stages together, and denoting the constraints (2.1b)-(2.1e) as $x \in X$ and the constraints (2.1h)-(2.1j) written for scenario s as $y^s \in Y^s$, the *deterministic equivalent form* for (2.1) is given as follows:

$$\text{Minimize} \quad cx + \sum_{s \in S} p_s g_s y^s \tag{2.2a}$$

$$\text{subject to} \quad T_s x + W_s y^s = r_s, \quad \forall s \in S \tag{2.2b}$$

$$x \in X, y^s \in Y^s, \quad \forall s \in S. \tag{2.2c}$$

Theoretically, any solution approach for general linear programming (LP) or mixed-integer programming (MIP) models are applicable to SPs with recourse when they are written in this aggregated form. However, the size of an SP model is comparable to the size of a typical LP or MIP model multiplied by the number of scenarios ($n_1 + n_2|S|$ variables and $m_1 + m_2|S|$ constraints), and if there exist a large number of scenarios, the size of the SP prohibits solving it without exploiting its inherent special structures. The representation (2.2b)–(2.2c) reveals a *dual angular structure* that lends stochastic programs well to decomposition schemes, and many algorithmic designs rely on this structure.

In the following, we first review solution approaches for two-stage stochastic linear programming (SLP) in Section 2.1.1. In Section 2.1.2, we survey different decomposition frameworks that have been used to solve stochastic mixed-integer programs (SMIPs), including decomposition by stage (primal decomposition), decomposition by scenario (dual decomposition), and test-set decomposition. In Section 2.1.3, we detail some convexification methods that have been used for solving mixed-integer 0-1 stochastic problems, particularly, employing disjunctive programming and Reformulation-Linearization Technique (RLT)-based cutting plane methods. In Section 2.1.4, we discuss three enumerative methods using tender variables when the technology matrix is fixed and the second-stage contains purely integer variables.

2.1.1 Solution Approaches for Two-Stage SLPs

Various specialized solution approaches have been proposed for two-stage SLP models. The most widely applied are decomposition based approaches starting from the L-shaped method by Van Slyke and Wets (1969), which is in essence Benders' decomposition (Benders (1962)). In this framework, for each first-stage solution $\hat{x}$ produced by a *master program*, one solves the corresponding second-stage problem based on (2.1f)–(2.1j) with x fixed at $\hat{x}$. In the

following, we will introduce the basic L-shaped method and then discuss its variations. For details of these methods, we refer interested readers to Kall and Wallace (1994) and Birge and Louveaux (1997). The structure and discussion in this subsection is primarily based on these two books and the original papers cited for each method.

The L-Shaped Method

The L-shaped method generates *feasibility cuts*, which is of special relevance to SPs that are not known to have (relatively) complete recourse, along with *optimality cuts*, by building an outer linearization of the recourse function using Benders' decomposition. (Alternatively, *inner linearization* methods such as Dantzig-Wolfe decomposition (Dantzig and Wolfe (1960)) can be applied to the dual.) The algorithm is described as follows:

Step 1. Solve the master problem:

$$\begin{aligned} \text{Minimize} \quad & c^T x + \eta && \text{(2.3a)} \\ \text{s.t.} \quad & x \in X, \eta \text{ unrestricted, and:} && \text{(2.3b)} \\ & \text{feasibility cuts obtained from Step 2 in previous iterations,} && \text{(2.3c)} \\ & \text{optimality cuts obtained from Step 3 in previous iterations,} && \text{(2.3d)} \end{aligned}$$

where η is a variable representing an approximation of the expected second-stage objective value. Obtain an optimal solution $(\hat{x}, \hat{\eta})$.

Step 2. Check the feasibility of the current solution $\hat{x}$ for all scenarios $s \in S$ using:

$$\begin{aligned} \text{Minimize} \quad & w_s = ev^+ + ev^- && \text{(2.4a)} \\ \text{s.t.} \quad & Wy + Iv^+ - Iv^- = r_s - T_s\hat{x}, && \text{(2.4b)} \\ & (y, v^+, v^-) \geq 0. && \text{(2.4c)} \end{aligned}$$

Evidently, this system is feasible and the objective is nonnegative for any given $\hat{x}$, and $\hat{x}$ is feasible to scenario s if and only if $w_s = 0$. Hence, if $w_s > 0$, then generate a *feasibility cut* $\hat{u}^T T_s x \geq \hat{u}^T r_s$, where $\hat{u}^T$ is an optimal solution to (2.4), and add it into (2.3c). Return to Step 1.

Step 3. Check the optimality of the current solution $(\hat{x}, \hat{\eta})$ for all scenarios $s \in S$ using:

$$\begin{aligned} \text{Maximize} \quad & r_s^T u_s - \hat{x}^T T_s^T u_s && \text{(2.5a)} \\ \text{s.t.} \quad & W^T u_s \leq g_s, && \text{(2.5b)} \\ & u \geq 0. && \text{(2.5c)} \end{aligned}$$

Let $\bar{u}_s$, $\forall s \in S$, represent optimal solutions to these problems. If $\hat{\eta} < \sum_{s \in S} p_s \hat{u}_s^T (r_s - T_s \hat{x})$, then add an optimality cut

$$\eta \geq \sum_{s \in S} p_s \hat{u}_s^T (r_s - T_s \hat{x})$$

to (2.3d) of the master problem (2.3), and return to Step 1; otherwise, stop with the solution $(\hat{x}, \hat{\eta})$ being optimal. □

Based on the above procedure, Birge and Louveaux (1988) have proposed a multicut L-shaped method that, at each iteration, adds one cut from each subproblem as necessary. The multicut version makes use of particular structures of the subproblems, and is expected to perform more effectively when the number of scenarios is not significantly larger than the number of original first-stage constraints.

However, the multicut version may have to add a large number of cuts in each iteration and thereby lead to a much larger master problem. Hence, some hybrid methods have been devised such as grouping the scenarios and adding one cut for each group of scenarios.

Regularized Decomposition

The regularized decomposition method due to Ruszczyński (1986, 1993) was developed to resolve some drawbacks of the L-shaped type of cutting plane approach. The initial iterations of cutting plane approaches are usually inefficient, because a good approximation of the piecewise linear convex function $f_s(x)$ is not yet available. When the approximation does not sufficiently reflect the shape of the function especially in the vicinity of optimal solutions, the solution tends to be unstable. Meanwhile, the number of cuts increases monotonically without a reliable deletion rule to control the problem size. The idea of regularized decomposition is to modify the master problem in such a way that, when starting with some overall feasible first-stage solution x_0, the next solution is not too different from x_0. For this purpose, the regularized decomposition method poses the problem as a non-smooth optimization problem, and adds a regularizing term $\alpha||x - x_0||^2$ in the objective function, where x_0 is the incumbent solution that is updated, as necessary, after generating an optimality cut.

Bunching and Trickling Down

Since we need to solve many similar structured second-stage subproblems when using decomposition methods, the motivation of *bunching* by Wets (1988) is to save computational effort by sharing some basis B among the subproblems. It applies when randomness appears only in the second-stage costs (g_s) or the second-stage RHS ($b_s \equiv T_s x - r_s$). Because the constraint system does not include any randomness, the optimal basis B for the second-stage dual subproblem is then *dual feasible* to all the second-stage subproblems. Therefore, any scenario also having B as a *primal feasible basis* can then be "bunched" together, and we can generate an optimality cut for each such bunch at Step 3 of the L-shaped method.

From one basis B, its neighbor can be obtained based on a single pivot using the dual simplex method. The neighboring relationship between the bases prompts a systematic way of searching for common bases. The key idea of *trickling down* is to build up a search tree structure that keeps track of the basis at each node, and each child node is generated by

performing one dual simplex pivot over the parent node, and is denoted as the i^{th} child of the parent node if the pivot is performed on the i^{th} row. Since different bases can share a common neighbor basis, a particular basis node may be generated from different parent nodes. Haugland and Wallace (1988) have shown that the efficacy of trickling down highly depends on the sequence in which the right-hand-sides are selected. In their method, the root node of the search tree is obtained from the median RHS, and later on, the values are selected by "smoothing" rules that prefer right-hand-sides that are closer to the median, and postpone selecting those that are further from the median. In this manner, the existing pivots in the tree can be utilized as much as possible to create a small and shallow tree. More importantly, when a node is examined, many children nodes may have to be generated before obtaining an optimal solution. Therefore, if the RHS selected each time is very different than those already studied when obtaining the current nodes in the tree, new pivots will have to be made to create a new branch, which leads to long paths and a deep tree that bears many inefficient pivots.

Sampling Method - Stochastic Decomposition

We have primarily focused on stochastic linear programming recourse models with discrete probabilities. As we have mentioned, continuous distributions can always be approximated using discrete ones. However, a good approximation depends on a large number of discretized points to represent the continuous distribution. When the problem has a non-trivial number of random parameters, a discretized approximation can lead to a tremendously large number of scenarios, or, second-stage subproblems, which can be computationally prohibitive. This motivates the many types of sampling methods for stochastic programs, such as the stochastic decomposition method due to Higle and Sen (1991), which is again based on the L-shaped method and assumes complete recourse and a known lower bound on the second-stage function value. In this method, one sample is randomly taken at each iteration, and a lower bound with respect to the expectation is produced. Because earlier cuts are based on fewer samples, the cuts generated gradually "phase out" as the iterations progress.

2.1.2 Stage, Scenario, and Test-Set Decomposition for Two-Stage SMIPs

Similar to the solution approaches for SLPs, those for SMIPs also heavily depend on decomposition. We discuss two major groups of decomposition methods for SMIPs—by stages and by scenarios—as well as a novel approach called test-set decomposition. Decomposition by stages, also known as primal decomposition, essentially adopts the framework of Benders' decomposition, or the L-shaped method. We discuss these methods in Section 2.1.2.1. Scenario decomposition methods, also known as dual decomposition, work with relaxing and restoring the *non-anticipativity condition*, which, in a two-stage problem setting, simply means that all scenario outcomes should be based on some identical first-stage solution. When this con-

dition is relaxed, smaller-scaled problems corresponding to scenarios are obtained that are easier to solve, but they also yield different first-stage solutions, which need to be reconciled. These methods are covered in Section 2.1.2.2. In Section 2.1.2.3, we describe the test-set decomposition approach by Hemmecke and Schultz (2003), which decomposes the problem's Graver test-sets, instead of the problem itself.

2.1.2.1 Stage Decomposition: L-Shaped Methods

The simplest form of stochastic mixed-integer programs contain purely binary first-stage variables and purely continuous second-stage variables. In this case, Benders' decomposition, or the L-shaped method, can be easily applied using some form of enumeration on the binary first-stage variables, as in Wollmer (1980). Alternatively, and also more generally, if the second-stage problems are easy to solve (not necessarily purely continuous), then these problems can be solved using the integer L-shaped method of Laporte and Louveaux (1993). This is a branch-and-cut (B&C) procedure that is implemented in the projected space of the first-stage variables. At each node, the two-stage problem is solved using Benders' decomposition, which generates feasibility and/or optimality cuts for the first-stage solution, as necessary. The Benders' *master problem* at some node q then takes the form:

$$\begin{aligned} \text{Minimize} \quad & cx + \eta && \text{(2.6a)} \\ \text{subject to} \quad & D_k x \geq d_k, \quad \forall k = 1, \ldots, F_q && \text{(2.6b)} \\ & L_k x \geq l_k, \quad \forall k = 1, \ldots, O_q && \text{(2.6c)} \\ & x \in X, \eta \text{ unrestricted}, && \text{(2.6d)} \end{aligned}$$

where $k = 1, \ldots, F_q$ and $k = 1, \ldots, O_q$ respectively index the feasibility cuts (2.6b) and the optimality cuts (2.6c) at node q. In the case of complete recourse or relatively complete recourse, feasibility cuts will not be generated from the second-stage problems, and (2.6b) will only include certain relevant constraints that fix particular components of x at 0 or 1 as per the branching restrictions. A nodal problem at node q in the B&C tree is re-solved whenever a feasibility or optimality cut is added. It is fathomed by the infeasibility of (2.6) or by the bounding processes, and is otherwise partitioned via branching if the solution to the continuous relaxation does not satisfy the binary restrictions. Unlike in a typical branch-and-bound (B&B) process, whenever integrality is satisfied and reveals a potentially better incumbent solution to the current (relaxed) master program, the node is not necessarily fathomed. Instead, the subproblem is solved to possibly generate an optimality cut.

Theoretically, this framework of embedding the L-shaped method in a B&C process applies to two-stage SMIPs in which the first stage contains binary or integer variables and valid optimality cuts are obtainable from the second-stage problems. In practice, however, valid optimality cuts are not easy to derive unless if the second stage contains purely continuous variables, or if the first stage contains purely binary variables. In the former case, the second-stage value functions $f_s(\cdot)$ are piecewise linear and convex, and so, valid optimality cuts can be derived using the dual variables directly. In the latter case, for any binary

feasible solution x^r, defining $I_r \equiv \{i \in I : \bar{x}_i^r = 1\}$, the following is a set of valid optimality cuts:

$$\eta \geq (Q(x^r) - L)[\sum_{i \in I_r} x_i - \sum_{i \notin I_r} x_i - |I_r| + 1] + L, \forall r = 1, \ldots, R, \tag{2.7}$$

where $Q(x) \equiv \sum_s p_s f_s(x)$, L is a lower bound on Q, and $1, \ldots, R$ are indices of all the binary feasible solutions that have been encountered thus far.

When the second-stage problems contain integer variables, their value functions $f_s(\cdot)$ are lower semi-continuous (Blair and Jeroslow (1982)) and in general nonconvex and discontinuous. Thus optimality cuts are not readily available from dual variables as in the continuous case. In such cases, when the two-stage program has a deterministic cost vector g and recourse matrix W, Carøe and Tind (1998) propose to use a subset $\mathcal{F}$ of the *dual price functions* of the second-stage integer programs (see Nemhauser and Wolsey (1999) for dual price functions for integer programs) to derive feasibility cuts and optimality cuts. These functions are, however, nonlinear in general, resulting in a nonlinear form of Problem (2.6). Moreover, the dual price function class $\mathcal{F}$ has to be sufficiently large so that the duality gap between the second-stage problem and its ($\mathcal{F}$-) dual is closed. The authors show that this is achieved when the second-stage problems are solved using cutting plane techniques or branch-and-bound (B&B). In these cases, finite termination is also established, given that (2.6) can be solved finitely. In particular, when Gomory's cuts are applied, the dual price functions can be transformed into linear functions having mixed-integer variables, and (2.6) is then a mixed-integer linear problem in lieu of a nonlinear problem.

The above decomposition methods work with the primal second-stage problems, given any fixed first-stage solution. Within this framework, alternative methods have been developed to obtain optimality cuts in the case of integer recourse. We will discuss some of these methods in Section 2.1.3. In Section 2.1.2.2 below, we turn to another type of decomposition approach that relaxes the non-anticipativity condition and works with copies of the first-stage solution to restore this condition.

2.1.2.2 Scenario Decomposition: Relaxation of the Non-anticipativity Condition

The second-stage problems are related to each other through the first-stage solution only. If we make a copy of the first-stage variable x for each scenario, say x^s for $s \in S$, and enforce the *non-anticipativity condition*, $x^1 = x^s$, $\forall s \in S$, conveniently written as $\sum_{s \in S} H^s x^s = 0$ for some suitable $H^s_{l \times n_1}$, then the deterministic equivalent problem shown in (2.2) can be rewritten as follows:

$$\min\{\sum_{s \in S} p_s(cx^s + g_s y^s) : (2.2\text{b}), (2.2\text{c}), \sum_{s \in S} H^s x^s = 0\}. \tag{2.8}$$

The Lagrangian dual of (2.8) is $\max_{\lambda \in \mathbb{R}^l} \{\sum_{s \in S} D_s(\lambda)\}$, where

$$D_s(\lambda) \equiv \min\{p_s(cx^s + g_s y^s) + \lambda(H^s x^s) : (2.2b), (2.2c)\}, \forall s \in S.$$

The Lagrangian dual value yields a tighter lower bound for Problem (2.2) than its LP-relaxation. Carøe and Schultz (1999) accordingly propose a dual decomposition algorithm that uses the values of the Lagrangian dual as lower bounds in a B&B process. The tradeoff here is having to solve a non-smooth Lagrangian dual problem compared to a linear program. At each nodal problem, after solving the associated Lagrangian dual problem, the solutions x^s, $\forall s \in S$, are averaged and rounded to obtain some $\bar{x}$ satisfying the integrality restrictions. The objective value of (2.2) obtained after fixing x at $\bar{x}$ is used to update the upper bound. At the branching step, for a selected branching variable x_i (component of x), constraints $x_i \leq \lfloor \bar{x}_i \rfloor$ and $x_i \geq \lceil \bar{x}_i \rceil$, or $x_i \leq \bar{x}_i - \epsilon$ and $x_i \geq \bar{x}_i + \epsilon$ for some tolerance $\epsilon > 0$, are applied at the two children nodes, respectively, depending on whether x_i is an integer or continuous variable. This algorithm is finitely convergent if X is bounded and the x variables are purely integer-restricted. Schultz and Tiedemann (2003) extend this approach to solve stochastic programs that include an additional objective term based on the probability of a risk function exceeding a pre-specified threshold value. A survey for SMIPs having risk measures can be found in Schultz (2003).

Alonso-Ayuso et al. (2003) also relax the non-anticipativity condition in their branch-and-fix coordination algorithm (BFC) for pure 0-1 multi-stage stochastic programs, or mixed 0-1 two-stage stochastic programs having purely binary variables in the first stage. They assume deterministic technology and recourse matrices. In this algorithm, both the non-anticipativity condition and the binary restrictions on x are relaxed, resulting in a linear problem for each scenario. A B&B tree is developed for each scenario at a terminal node of the scenario tree, and the branch and fix operations are performed on the so-called "twin node families". In a two-stage setting, a twin node family includes the nodal problems in the B&B trees for all scenarios that have the already fixed x-variables equal to a same value (0 or 1). To enforce the non-anticipativity condition, the same branching variable is selected for the active nodes in a twin node family. Note that these nodes belong to different scenarios. Two new twin node families are then formed by fixing the selected branching variable at 0 and 1, respectively. Lower bounds on the objective value are updated by calculating the deterioration of the objective value due to the fixing.

2.1.2.3 Decomposition of Test-Sets

When the cost vector g, the technology matrix T, and the recourse matrix W are all deterministic, and randomness only appears in the right-hand-side values r_s, Hemmecke and Schultz (2003) take advantage of the dual angular structure in (2.2) to develop a test-set decomposition based approach for two-stage stochastic integer programs. If the problem contains continuous variables, certain inherited computational difficulties from such an approach

demand extra care.

A finite *universal test-set* for a family of integer problems

$$(IP)_{c,b}: \quad \min\{cz : Az = b, z \in \mathbb{Z}^d_+\}, \tag{2.9}$$

where $A \in \mathbb{Q}^{l\times d}$ is fixed, and where $c \in \mathbb{R}^d$ and $b \in \mathbb{R}^l$ vary, contains a set of vectors that can be used to solve any problem in this family $(IP)_{c,b}$ for a given c and b (Hemmecke (2000)). In this process, an initial feasible solution is first found using the test-set vectors, then an optimal solution is obtained by searching along improving directions, again using the vectors (called improving vectors) in the test-set. The IP Graver test-set (Graver (1975)) is one such finite universal test-set, and can be computed using the kernel of A, $ker(A)$, as shown in Hemmecke (2000).

A major drawback of using the test-set approach to solve integer programs is that a large number of vectors need to be stored, even for small-sized problems. Hence, for the typically large-sized stochastic integer programs, directly applying this approach is impractical. Exploring the dual angular structure of (2.2), Hemmecke and Schultz (2003) show that Graver test-set vectors for this type of problem can be decomposed into, and then constructed from, a small number of building-block vectors, and the latter can then be used to solve large-scaled stochastic integer programs.

Let

$$A_{|S|} \equiv \begin{pmatrix} A & 0 & 0 & \dots & 0 \\ T & W & 0 & \dots & 0 \\ T & 0 & W & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T & 0 & 0 & \dots & W \end{pmatrix} \text{ and } A_1 \equiv \begin{pmatrix} A & 0 \\ T & W \end{pmatrix}. \tag{2.10}$$

Observing that $(u, v_1, \dots, v_{|S|}) \in ker(A_{|S|}) \Leftrightarrow (u, v_1), \dots, (u, v_{|S|}) \in ker(A_1)$, Hemmecke and Schultz (2003) propose to use the individual vectors $u, v_1, \dots,$ and $v_{|S|}$ as building blocks of the vectors $(u, v_1, \dots, v_{|S|})$ in the Graver test-set of $A_{|S|}$. These building blocks, in lieu of the test-set vectors, are collected into a set $\mathcal{H}_\infty$ and arranged in pairs (u, V_u), where V_u is the set of vectors v such that $(u, v) \in ker(A)$. The set of building blocks $\mathcal{H}_\infty$ is shown to be finite, and can be computed using a finite algorithm. Although the computation of $\mathcal{H}_\infty$ is expensive, since it depends only on A, W, and T, and is independent of $|S|$, cost coefficients, and right-hand-side values, the proposed approach is insensitive to the number of scenarios once $\mathcal{H}_\infty$ is obtained. After computing and storing the building blocks, the test-set vectors are constructed during the two steps of obtaining an initial feasible solution and finding improving directions to solve (2.2). The finiteness of $\mathcal{H}_\infty$ guarantees the finiteness of the proposed algorithm.

2.1.3 Disjunctive Programming and RLT Cutting Plane Methods for Mixed 0-1 Stochastic Programs

In this section, we assume the following:

A1. No general integer variable exists in either stage, that is, $I_{int} = J_{int} = \emptyset$.

A2. The continuous variables in both stages are bounded. Moreover, for the purpose of exposition, the continuous variables in the second stage are scaled onto [0,1], with the corresponding bounding restrictions $y \leq 1$ being absorbed within (or implied by) (2.1g).

A3. For any feasible first-stage variable $x \in X$, the second-stage problems are feasible (relative complete recourse).

We study two groups of cutting plane approaches for solving two-stage SMIPs having 0-1 mixed-integer variables, and focus on the idea of sharing cut coefficients among scenarios. The first group is based on disjunctive programming, and includes the papers by Carøe and Tind (1997), Sen and Higle (2005), Sen and Sherali (2006), and Ntaimo and Sen (2005). The second group applies the Reformulation-Linearization Technique (RLT) to derive cutting planes, and is represented by the research of Sherali and Fraticelli (2002). Both disjunctive cuts and RLT cuts are convexification cutting planes applied to deal with the binary variables in the second-stage problems. In solving stochastic programs, particularly when the number of scenarios is large, memory space is crucial. If some cut coefficient can be shared between scenarios or iterations, we can gain much in memory savings and algorithmic efficacy. In Section 2.1.3.1, we therefore focus on the convexification process and cut-sharing properties for solving mixed-integer 0-1 second-stage problems. In Section 2.1.3.2, we then discuss a solution approach for problems containing purely continuous first-stage variables, as continuous first-stage variables pose an extra challenge on assuring convergence. In Chapter 3, when we present algorithms for two-stage stochastic programs containing 0-1 mixed-integer variables in both stages, we will relate the RLT cuts generated therein with the cuts generated using disjunctive programming to show their inter-connections.

2.1.3.1 Solving Mixed-Integer 0-1 Second-Stage Problems

Given an index set H and some polyhedral sets $\mathcal{P}_h$, for $h \in H$, a disjunctive set $\mathcal{P} = \bigcup_{h \in H} \mathcal{P}_h$ is said to be in *disjunctive normal form*. Using disjunctive programming, we can characterize the closure of the convex hull of $\mathcal{P}$ and generate valid inequalities and even facets for this representation (see, for example, Blair and Jeroslow (1978), Balas (1979), and Sherali and Shetty (1980)). The class of 0-1 mixed-integer programs (MIP) belong to a special type of disjunctive programs called *facial disjunctive programs*, written in *conjunctive normal form*, i.e., a conjunction over $i \in I_b$ of the disjunction $\{x_i = 0 \vee x_i = 1\}$. For this type of programs,

we can obtain the convex hull of the feasible solution set using a sequential convexification process.

The lift-and-project algorithm proposed by Balas et al. (1993) solves 0-1 mixed-integer programs using disjunctive programming. Carøe and Tind (1997) modified this approach and applied it to the deterministic equivalent form of SMIP (Problem (2.2)) in which the first-stage variables are purely continuous (i.e., $I_b = \emptyset$) and the second stage contains both continuous and binary variables. Let $\mathcal{P}^s$ be the set of solutions in the space of $(x, y^h, \forall h \in S)$ when only the constraints on x and y^s are considered. A direct application of the lift-and-project method would generate cuts in the $(x, y^h, \forall h \in S)$ space, sequentially treating a single variable y^s_j for $j \in J_b$ and $s \in S$ as binary, and the remaining y-variables as continuous. Exploring the dual angular structure of the deterministic equivalent form, Carøe and Tind (1997) proposed the generation of cuts for $\mathcal{P}^s$ in the (x, y^s) space for each scenario $s \in S$. This is valid because all the y^s, $s \in S$, are independent, and by the assumption of relatively complete recourse, the projection of $\mathcal{P}^s$ and the projection of the original solution set on the (x, y^s) space are the same.

To generate such cuts for a current solution $(\bar{x}, \bar{y}^s)$ for which the value for $\bar{y}^s_q$, $q \in J_b$, is fractional, consider the disjunction $\mathcal{P}^s = \mathcal{P}^s_0 \cup \mathcal{P}^s_1$, where

$$\begin{aligned}
\mathcal{P}^s_0 = \{ & Ax \geq b && \leftarrow \tau_0 && (2.11a)\\
& T_s x + W_s y^s \geq r_s && \leftarrow \lambda_0 && (2.11b)\\
& -y^s_q \geq 0\} && \leftarrow \varsigma_0 && (2.11c)
\end{aligned}$$

and

$$\begin{aligned}
\mathcal{P}^s_1 = \{ & Ax \geq b && \leftarrow \tau_1 && (2.11d)\\
& T_s x + W_s y^s \geq r_s && \leftarrow \lambda_1 && (2.11e)\\
& y^s_q \geq 1\} && \leftarrow \varsigma_1 && (2.11f)
\end{aligned}$$

with the associated multipliers as shown above, and where we assume that the nonnegativity and the bounding constraints for x and y are absorbed in $Ax \geq b$ and $T_s x + W_s y^s \geq r_s$, respectively. Define e_q as the q^{th} unit row vector. A convexification cutting plane for the disjunction (2.11) is generated by solving the following linear program:

$$\begin{aligned}
\text{Minimize} \quad & \alpha\bar{x} + \beta\bar{y}^s - \gamma && && (2.12a)\\
\text{subject to:} \quad & \alpha - \tau_h A - \lambda_h T_s \geq 0, && \forall h = 0, 1 && (2.12b)\\
& \beta + \varsigma_0 e_q - \lambda_0 W_s \geq 0 && && (2.12c)\\
& \beta - \varsigma_1 e_q - \lambda_1 W_s \geq 0 && && (2.12d)\\
& \tau_0 b + \lambda_0 r_s - \gamma \geq 0 && && (2.12e)\\
& \varsigma_1 + \tau_1 b + \lambda_1 r_s - \gamma \geq 0 && && (2.12f)\\
& \alpha\bar{x} + \beta\bar{y}^s - \gamma \geq -1 && && (2.12g)\\
& \tau, \lambda, \varsigma \geq 0, \ \alpha, \beta, \gamma \text{ unrestricted.} && && (2.12h)
\end{aligned}$$

Let $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\lambda}, \bar{\tau}, \bar{\varsigma})$ be an optimal solution of (2.12). If the objective value obtained for (2.12a) is negative (i.e., equals -1 due to (2.12g)), the inequality

$$\bar{\alpha}x + \bar{\beta}y^s \geq \bar{\gamma} \tag{2.13}$$

eliminates the current fractional solution.

When using the L-shaped method (or Benders' decomposition) to solve two-stage stochastic programs, we need the value functions of the Benders' subproblems (i.e., the second-stage problems) to be convex. However, if the second-stage problems contain binary or integer variables, their value functions are in general nonconvex and discontinuous. This then requires a relaxation of the binary restrictions in the second-stage problems and an accompanying convexification process.

When solving the second-stage problems, a given solution $\bar{x}$ is available from the first-stage problem and becomes part of the right-hand-sides of the second-stage problems. Valid inequalities obtained in this context will be in the form of $\bar{\beta}_s y^s \geq \bar{\gamma}_s$. It is helpful to lift these (possibly facetial) cuts to be valid in the (x, y^s) space, in the form of $\alpha_s x + \bar{\beta}_s y^s \geq \bar{\gamma}_s + \alpha_s \bar{x}$. By treating x as a variable, these cuts can be reused in subsequent Benders' iterations by updating the first-stage solutions from the Benders' master problem, as detailed in Sherali and Fraticelli (2002). This cut lifting is not well-posed when x is continuous. If x is restricted to be binary, then the feasible x-solutions are always facial to the convex hull of the two-stage constraint set, and we can lift a valid inequality using the disjunction

$$\left\{ \begin{array}{l} Ax \geq b \\ T_s x + W_s y^s \geq r_s \\ -x_i \geq 0 \end{array} \right\} \bigvee \left\{ \begin{array}{l} Ax \geq b \\ T_s x + W_s y^s \geq r_s \\ x_i \geq 1 \end{array} \right\}. \tag{2.14}$$

The lifted cut can be obtained via the solution to the following linear program, where $\bar{\beta}_s$ and $\bar{\gamma}_s$ are cut coefficients already obtained in the y^s space:

$$\begin{aligned}
\text{Minimize} \quad & \alpha_s \bar{x} && \text{(2.15a)} \\
\text{subject to:} \quad & \alpha_s + \varsigma_0 e_i - \tau_0 A - \lambda_0 T_s \geq 0 && \text{(2.15b)} \\
& \alpha_s - \varsigma_1 e_i - \tau_1 A - \lambda_1 T_s \geq 0 && \text{(2.15c)} \\
& -\lambda_h W_s \geq -\bar{\beta}_s, \quad \forall h = 0, 1 && \text{(2.15d)} \\
& -\alpha_s \bar{x} + \tau_0 b + \lambda_0 r_s \geq \bar{\gamma}_s && \text{(2.15e)} \\
& -\alpha_s \bar{x} + \varsigma_1 + \tau_1 b + \lambda_1 r_s \geq \bar{\gamma}_s && \text{(2.15f)} \\
& \alpha_s \bar{x} \geq -1 && \text{(2.15g)} \\
& \tau, \lambda, \varsigma \geq 0, \ \alpha_s \text{ unrestricted.} && \text{(2.15h)}
\end{aligned}$$

Also, in the purely binary first-stage problem setting, if additionally the recourse matrix is fixed, Sen and Higle (2005) propose to lift cuts for second-stage problems in the (x, y^s)-space, such that the cuts share the same coefficients $\bar{\beta}$ for the y-variables. These cuts are

then in the form of $\bar{\beta}y^s \geq \gamma_s - \alpha_s x$. This property is named as Common Cut Coefficients (C^3), and is established as follows.

Given a first-stage solution $\bar{x}$ and a fixed recourse matrix W, consider the disjunction

$$\left\{ \begin{array}{l} Wy^s \geq r_s - T_s\bar{x} \\ -y^s_q \geq 0 \end{array} \right\} \bigvee \left\{ \begin{array}{l} Wy^s \geq r_s - T_s\bar{x} \\ y^s_q \geq 1 \end{array} \right\}. \tag{2.16}$$

The common cut coefficients $\bar{\beta}$ are obtained via the optimal value for β given by the following linear program, where $\bar{y}^s$, $\forall s \in S$, are the current second-stage solutions:

$$\begin{aligned}
\text{Minimize} \quad & \sum_s p_s(\beta\bar{y}^s - \pi_s) && \text{(2.17a)}\\
\text{subject to:} \quad & \beta + \varsigma_0 e_q - \lambda_0 W \geq 0 && \text{(2.17b)}\\
& \beta - \varsigma_1 e_q - \lambda_1 W \geq 0 && \text{(2.17c)}\\
& \lambda_0(r_s - T_s\bar{x}) - \pi_s \geq 0, \quad \forall s \in S && \text{(2.17d)}\\
& \varsigma_1 + \lambda_1(r_s - T_s\bar{x}) - \pi_s \geq 0, \quad \forall s \in S && \text{(2.17e)}\\
& \sum_s p_s(\beta\bar{y}^s - \pi_s) \geq -1, && \text{(2.17f)}\\
& \pi, \lambda, \varsigma \geq 0, \ \beta \text{ unrestricted.} && \text{(2.17g)}
\end{aligned}$$

Letting $\bar{\lambda}$ be an optimal value for λ in Problem (2.17), the cut

$$\bar{\beta}y \geq \min\{\bar{\lambda}_0 r_s - \bar{\lambda}_0 T_s x, \bar{\lambda}_1 r_s + \varsigma_1 - \bar{\lambda}_1 T_s x\} \tag{2.18}$$

is then valid for scenario s, and its piecewise linear concave right-hand-side can be convexified using its epigraph over the bounded region $\bar{X} \equiv \{x \in \mathbb{R}^{n_1}_+ | Ax \geq b\}$. Denote this epigraph as

$$\Pi^s_{\bar{X}} \equiv \{(\varpi, x) | x \in \bar{X}, \varpi \geq \mathcal{R}^s(x)\}, \tag{2.19}$$

where $\mathcal{R}^s(x) \equiv \min\{\bar{\lambda}_0 r_s - \bar{\lambda}_0 T_s x, \bar{\lambda}_1 r_s + \varsigma_1 - \bar{\lambda}_1 T_s x\}$.

In this process, for $x \in \bar{X}$, we can find a lower bound for $\mathcal{R}^s(x)$, say, $\mathcal{L}$. Each affine function in $\mathcal{R}^s(x)$ is then translated by $\mathcal{L}$, if necessary, to ensure that $\varpi \geq 0$. This will translate the convex hull of the epigraph as well. After convexification cuts are obtained, they can be translated back by $-\mathcal{L}$ to recover the original values. So without loss of generality, assume that $\mathcal{L} = 0$.

We can then represent $\Pi^s_{\bar{X}}$ as the following disjunction

$$\left\{ \begin{array}{l} \varpi \geq \bar{\lambda}_0 r_s - \bar{\lambda}_0 T_s x \\ Ax \geq b \end{array} \right\} \bigvee \left\{ \begin{array}{l} \varpi \geq \bar{\lambda}_1 r_s + \bar{\varsigma_1} - \bar{\lambda}_1 T_s x \\ Ax \geq b \end{array} \right\}. \tag{2.20}$$

The right-hand-side of (2.18) can then be convexified based on this disjunction by solving the following linear program for each scenario s.

$$\begin{aligned}
\text{Minimize} \quad & \sigma_s \bar{x} + \upsilon_s - \delta_s && \text{(2.21a)}\\
\text{subject to:} \quad & \sigma_s - \tau_h A - \theta_h(\bar{\lambda}_h T_s) \geq 0, \quad \forall h = 0, 1 && \text{(2.21b)}\\
& \upsilon_s - \theta_h \geq 0, \quad \forall h = 0, 1 && \text{(2.21c)}\\
& \tau_0 b + \theta_0(\bar{\lambda}_0 r_s) - \delta_s \geq 0 && \text{(2.21d)}\\
& \tau_1 b + \theta_1(\bar{\lambda}_1 r_s) + \theta_1 \bar{\varsigma}_1 - \delta_s \geq 0 && \text{(2.21e)}\\
& \theta_0 + \theta_1 = 1 && \text{(2.21f)}\\
& \theta, \tau \geq 0 \ \sigma, \delta, \upsilon \text{ unrestricted.} && \text{(2.21g)}
\end{aligned}$$

Note that from (2.21c) and (2.21f), we have that $\upsilon_s > 0$. For an optimal extreme point solution $(\bar{\sigma}_s, \bar{\upsilon}_s, \bar{\delta}_s)$, we then have $clconv(\Pi^s_{\bar{X}}) = \{(\varpi, x) | x \in \bar{X}, \varpi \geq (\bar{\delta}_s/\bar{\upsilon}_s) - (\bar{\sigma}_s/\bar{\upsilon}_s)x\}$. This completes the convexification of the piecewise linear concave $\mathcal{R}^s(x)$. Letting $\bar{\alpha}_s = \bar{\sigma}_s/\bar{\upsilon}_s$ and $\bar{\gamma}_s = \bar{\delta}_s/\bar{\upsilon}_s$, the inequality

$$\bar{\beta} y \geq \bar{\gamma}_s - \bar{\alpha}_s x \tag{2.22}$$

is then valid for scenario s, $\forall s \in S$, and all scenarios share the same coefficient $\bar{\beta}$ for y in the new cuts, which need to be stored only once.

The disjunctive decomposition (D^2) method by Sen and Higle (2005) thus applies Benders decomposition to solve two-stage stochastic programs, using the above C^3-scheme embodied by (2.17) and (2.21) to solve the second-stage subproblems.

The two approaches by Carøe and Tind (1997) and Sen and Higle (2005) directly stem from disjunctive programming. Using another approach, the Reformulation-Linearization Technique (RLT) (see Sherali and Adams (1990, 1994)), we can sequentially construct (partial) convex hulls for the second-stage subproblems. Sherali and Fraticelli (2002) proposed a method for solving two-stage problems having purely binary first-stage variables using a modified Benders' decomposition scheme, where the subproblems are solved by adding RLT cuts generated in the (x, y^s) space. The idea is to store the cuts as functions of x, so that at each Benders' iteration, when new x-values are obtained from the master (first-stage) problem, the cutting planes obtained previously from the second-stage subproblem solutions can be reused in the subsequent subproblems of the same scenario simply by updating the x-values.

In a Benders' decomposition setting, each second-stage problem is solved independently; hence, we omit the superscript s for y-variables when no confusion arises. Further, denote the q^{th} column of matrix W_s as W^q_s, and the matrix formed by the remaining columns of W_s as $\tilde{W}_s$. Correspondingly, denote variable y without the q^{th} element as $\tilde{y}$ and let $\tilde{\beta}$ be its associated coefficient, i.e., $\beta y \equiv \tilde{\beta}\tilde{y} + \beta_q y_q$. To derive cuts in the (x, y) space, that is, to introduce x as a variable into the cut generation process, we include the bounding constraint $0 \leq x \leq e$ into the second stage. By the RLT process, given a solution $(\bar{x}, \bar{y})$, which has $\bar{y}_q$ fractional for some $q \in J_b$, we multiply $0 \leq x \leq e$ and the constraints in the second stage

by the factors y_q and $(1-y_q)$ to obtain a system in a higher dimensional space (x, y, z^x, z^y), including the new RLT variables z^x and z^y to represent the resulting nonlinear products, i.e.,

$$z^x \equiv xy_q \text{ and } z^y \equiv \tilde{y}y_q. \tag{2.23}$$

Denote the resulting system as:

$$T_s z^x + \tilde{W}_s z^y \geq (r_s - W_s^q)y_q \qquad \leftarrow \phi_0 \tag{2.24a}$$
$$-T_s z^x - \tilde{W}_s z^y \geq r_s - T_s x - \tilde{W}_s \tilde{y} - r_s y_q \qquad \leftarrow \phi_1 \tag{2.24b}$$
$$\Gamma_z z^x \geq h_b - \Gamma_x x - \Gamma_y y_q, \qquad \leftarrow \phi_b. \tag{2.24c}$$

The constraints (2.24a) and (2.24b) are obtained by multiplying (2.1g) (for scenario s) by y_q and $(1-y_q)$, respectively. The constraints (2.24c) are obtained by multiplying the bounding constraints of $0 \leq x \leq e$ by y_q and $(1-y_q)$, where Γ_z, Γ_x, and Γ_y are used to denote the resulting coefficient matrices for z^x, x, and y_q, respectively. Associating the dual multipliers ϕ as indicated in (2.24), the solution $(\bar{\alpha}_s, \bar{\beta}_s, \bar{\gamma}_s)$ of the following linear program yields the valid inequality $\bar{\beta}_s y \geq \bar{\gamma}_s - \bar{\alpha}_s x$ for scenario s in the projected space of the original variables (x, y):

$$\text{Minimize} \quad \tilde{\beta}_s \tilde{\tilde{y}} + \beta_{qs}\bar{y}_q + \alpha_s \bar{x} - \gamma_s \tag{2.25a}$$
$$\text{subject to:} \quad (\phi_0 - \phi_1)T_s + \phi_b \Gamma_z = 0 \tag{2.25b}$$
$$(\phi_0 - \phi_1)\tilde{W}_s = 0 \tag{2.25c}$$
$$\alpha_s = \phi_1 T_s + \phi_b \Gamma_x \tag{2.25d}$$
$$\tilde{\beta}_s = \phi_1 \tilde{W}_s \tag{2.25e}$$
$$\beta_{qs} = \phi_0(W_s^q - r_s) + \phi_1 r_s + \phi_b \Gamma_y \tag{2.25f}$$
$$\gamma_s = \phi_1 r_s + \phi_b h_b \tag{2.25g}$$
$$\tilde{\beta}_s \tilde{\tilde{y}} + \beta_{qs}\bar{y}_q + \alpha_s \bar{x} - \gamma_s \geq -1 \tag{2.25h}$$
$$\phi \geq 0, \ \alpha_s, \beta_s, \gamma_s \text{ unrestricted.} \tag{2.25i}$$

In a Benders' decomposition context where subproblems are solved using a cutting plane method, and where these cutting planes are valid in the (x, y)-space for $conv\{(x, y) \mid T_s x + Wy \geq r_s,\ 0 \leq x \leq e,\ y \geq 0,\ y_j \in \{0, 1\},\ \forall j \in J_b\}$, we can append these cuts along with any possibly added bounding constraints for the y-variables to $T_s x + Wy \geq r_s$, and Sherali and Fraticelli (2002) have shown that Benders' cuts generated using the dual solution of the augmented LP relaxation system are valid optimality cuts in terms of the first-stage variables.

Using this idea and disjunctive programming, Sen and Sherali (2006) develop a disjunctive decomposition-based branch-and-cut (D^2-BAC) approach, where subproblems are partially solved using branch-and-cut, and where optimality cuts applied in the first-stage master problem are generated using the disjunctive programming concept. In the B&B tree

for any subproblem in this process, each node is an LP relaxation of the subproblem that includes some node-dependent bounding constraints for y. Assuming that all the nodes are associated with feasible LP relaxations and are fathomed only by the bound computations, then at least one terminal node corresponds to an optimal solution. Let Q_s denote the set of terminal nodes of the tree that have been explored for the subproblem for scenario s, and let z_{qls} and z_{qus} denote the lower and upper bounds for y in the nodal subproblem q, for $q \in Q_s$. Let $\bar{\lambda}_{qs}$, $\bar{\psi}_{qls}$, and $\bar{\psi}_{qus}$ be dual solutions associated with the constraint set $Wy \geq r_s - T_s x$ and the lower and upper bounding constraints for y, respectively, in the nodal subproblem for node q. We then obtain the following disjunction:

$$\eta \geq \bar{\lambda}_{qs}[r_s - T_s x] + \bar{\psi}_{qls} z_{qls} - \bar{\psi}_{qus} z_{qus} \text{ for at least one } q \in Q_s. \tag{2.26}$$

Similar to the convexification process for (2.18), we can use the following disjunction to generate a disjunctive cut for (2.26):

$$\bigvee_{q \in Q_s} \left\{ \begin{array}{l} \eta + \bar{\lambda}_{qs} T_s x \geq \bar{\lambda}_{qs} r_s + \bar{\psi}_{qls} z_{qls} - \bar{\psi}_{qus} z_{qus} \\ Ax \geq b \end{array} \right\}. \tag{2.27}$$

Denote $\bar{\eta}$ as a current estimate lower bound for the second-stage value function, possibly obtained from previous iterations, and let $(\bar{\sigma}_s, \bar{\upsilon}_s, \bar{\delta}_s)$ be an optimal extreme point solution of the following linear program:

$$\begin{aligned}
\text{Minimize} \quad & \sigma_s \bar{x} + \upsilon_s \bar{\eta} - \delta_s && \text{(2.28a)} \\
\text{subject to:} \quad & \sigma_s - \tau_q A - \theta_q \bar{\lambda}_{qs} T_s \geq 0, \quad \forall q \in Q_s && \text{(2.28b)} \\
& \upsilon_s - \theta_q \geq 0, \quad \forall q \in Q_s && \text{(2.28c)} \\
& \tau_q b + \theta_q (\bar{\lambda}_{qs} r_s + \bar{\psi}_{qls} z_{qls} - \bar{\psi}_{qus} z_{qus}) - \delta_s \geq 0, \quad \forall q \in Q_s && \text{(2.28d)} \\
& \sum_{q \in Q_s} \theta_q = 1 && \text{(2.28e)} \\
& \theta, \tau \geq 0 \ \sigma, \delta, \upsilon \text{ unrestricted.} && \text{(2.28f)}
\end{aligned}$$

Again, due to (2.28c) and (2.28e), we have that $\bar{\upsilon}_s > 0$. A disjunctive cut that provides a lower bound on the second-stage value function can then be obtained in the form of $\eta_s \geq \bar{\gamma}_s - \bar{\alpha}_s x$, where $\bar{\alpha}_s = \bar{\sigma}_s / \bar{\upsilon}_s$ and $\bar{\gamma}_s = \bar{\delta}_s / \bar{\upsilon}_s$.

2.1.3.2 Pure Continuous and Mixed 0-1 First-Stage Problems

Following the cutting plane game concept of Jeroslow (1980), the disjunctive and RLT cut generation processes finitely solve the second-stage subproblems. Therefore, finite convergence of the D^2 algorithm of Sen and Higle (2005), and of the modified Benders' decomposition algorithm of Sherali and Fraticelli (2002), is afforded by the finite number of feasible first-stage solutions and by the finite cutting plane generation processes for solving subproblems. If the first stage contains continuous variables, however, a feasible first-stage

solution $\bar{x}$ may not be facial with respect to its bounding constraints, and the algorithms of Section 2.1.3.1 would no longer assure convergence.

To retain the facial property of the first-stage solutions, Ntaimo and Sen (2005) propose to build a B&B tree via a partitioning process in the projected space of the bounded first-stage variables so as to ultimately induce the aforementioned facial property.

Using the D^2 algorithm, when $\bar{x}$ is at a vertex of its bounding region, for the right-hand-side of the cut (2.22) generated by (2.17) and (2.21), we will have

$$\bar{\gamma}_s - \bar{\alpha}_s x = \min\{\bar{\lambda}_0 r_s - \bar{\lambda}_0 T_s x, \bar{\lambda}_1 r_s + \varsigma_1 - \bar{\lambda}_1 T_s x\} (\equiv \mathcal{R}^s(x)); \tag{2.29}$$

otherwise, this equality may be violated. Ntaimo and Sen (2005) then propose a finitely convergent D^2-based branch-and-cut (D^2-CBAC) algorithm for problems containing purely continuous first-stage variables. This algorithm constructs a B&B tree defined on the first-stage feasible region, applies the D^2 algorithm at each node, selects some scenario s and iteration k such that (2.29) is mostly violated, and partitions based on the disjunction prompted by (2.29), enforcing $\bar{\lambda}_0 r_s - \bar{\lambda}_0 T_s x \geq \bar{\lambda}_1 r_s + \varsigma_1 - \bar{\lambda}_1 T_s x$ on one child node and $\bar{\lambda}_0 r_s - \bar{\lambda}_0 T_s x \leq \bar{\lambda}_1 r_s + \varsigma_1 - \bar{\lambda}_1 T_s x$ on the other. The convexification cut (2.22) at the parent node is accordingly updated to $\bar{\beta} y \geq \bar{\lambda}_0^k r_s^k - \bar{\lambda}_0^k T_s^k x$ and $\bar{\beta} y \geq \bar{\lambda}_1^k r_s + \varsigma_1^k - \bar{\lambda}_1^k T_s^k x$, at the two children nodes, respectively.

Because there are finitely many disjunctive variables in the second stage, for some iteration of the embedded D^2 algorithm, there are a finite number of right-hand-sides of (2.18) that can be constructed; hence, there are finitely many partitions of the first-stage feasible region to consider, which leads to finite convergence of the D^2-CBAC algorithm.

If the first stage further contains both continuous and binary variables, we will propose a decomposition-based B&B (DBAB) algorithm that is guaranteed to converge to an optimal solution in Chapter 3.

2.1.4 Structural Enumeration Using a Fixed Technology Matrix

In this section, we assume the following:

A1. The first-stage feasible region X is bounded, and thus compact.

A2. For any $x \in X$, the second-stage problems are feasible (relative complete recourse).

A3. For any $x \in X$, the second-stage value functions are bounded.

A4. The second-stage variables are purely integer, that is, $y \in \mathbb{Z}_+^{n_2}$.

A5. The technology matrix T is fixed (i.e., deterministic).

A6. All elements in the recourse matrices W_s are integral. (Rational elements can be scaled to obtain integral elements.)

Assumption A1 assures finite termination for the enumeration schemes used in the algorithms in this section. Assumptions A2 and A3 imply that $f_s(x)$ and $u(r_s - Tx)$, $\forall u \in \mathbb{R}_+^{m_2}$, are finite. Using Assumption A5, we can transform the second-stage value functions as defined on the space of the so-called "tender variables", $-Tx$, which we denote as χ. That is,

$$F_s(\chi) = \min\{g_s y | W_s y \geq r_s + \chi, y \in \mathbb{Z}_+^{n_2}\} = f_s(x). \tag{2.30}$$

From Assumptions A4 and A6, we have that $W_s y \geq t$ implies $W_s y \geq \lceil t \rceil$, where $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ respectively denote the componentwise rounded-up and rounded-down forms of a vector. Therefore, we have that $f_s(x)$ is constant on the sets

$$\{x \in \mathbb{R}_+^{n_1} : \lceil r_s - Tx \rceil = \kappa\} = \{x : \kappa - r_s - e < -Tx \leq \kappa - r_s\}, \quad \forall \kappa \in \mathbb{Z}^{m_2}, \tag{2.31a}$$

and $F_s(\chi)$ is constant on the hyperrectangles

$$\prod_{j=1}^{m_2} (\kappa_j - r_{sj} - 1, \kappa_j - r_{sj}], \quad \forall \kappa \in \mathbb{Z}^{m_2}. \tag{2.31b}$$

Therefore, for some integral vector $k \equiv (k_{11}, \ldots, k_{sj}, \ldots, k_{|S|m_2})^T \in \mathbb{Z}^{|S|m_2}$, the expectation $Q(x) \equiv \sum_s p_s f_s(x)$ is constant over the intersection of the sets (2.31a):

$$C(k) \equiv \bigcap_{s \in S} \bigcap_{j=1}^{m_2} \{x : k_{sj} - r_{sj} - 1 < -T_j x \leq k_{sj} - r_{sj}\}, \quad \forall k \in \mathbb{Z}^{|S|m_2} \tag{2.32a}$$

and $\mathcal{Q}(\chi) \equiv \sum_s p_s F_s(\chi)$ is constant over the intersection of the hyperrectangles (2.31b):

$$\mathcal{C}(k) \equiv \bigcap_{s \in S} \prod_{j=1}^{m_2} (k_{sj} - r_{sj} - 1, k_{sj} - r_{sj}], \quad \forall k \in \mathbb{Z}^{|S|m_2}. \tag{2.32b}$$

Based on the above observation that function values are constant over regions, Schultz et al. (1998) and Ahmed et al. (2004) developed algorithms that construct partitions of the first-stage feasible region using $C(\cdot)$ and $\mathcal{C}(\cdot)$, respectively, and evaluate expected values, $Q(\cdot)$ and $\mathcal{Q}(\cdot)$, on these partitions. The algorithm in the former paper operates over the x-space, while that in the latter works in the χ-space.

Under Assumption A1 that the first-stage feasible region X is compact, Schultz et al. (1998) show that there are finite number of vertices of the sets $C(\cdot) \cap X$, and these vertices contain an optimal solution for x. (If X is not compact, this vertex set is only countable, and a level set is used to bound the feasible region to guarantee finite termination.) Aside from A1-A6, Schultz et al. (1998) further assume the following:

A7. The recourse matrix W is fixed.

A8. The first-stage variables are continuous, that is, $I_b = I_{int} = \emptyset$.

The purpose of these assumptions is to use the Gröbner-basis method from computational algebra to evaluate the second-stage value functions. Although the computation of Gröbner bases is expensive, it only uses the T and W matrices, and does not rely on the right-hand-side values. Hence, for fixed T and W, Gröbner bases need to be computed only once, and then for each different right-hand-side value, the second-stage function evaluation, which reduces to a single generalized division, is very easy. Using this idea, the objective value $cx + Q(x)$ is evaluated at each candidate point from the finite vertex set using Gröbner bases, and after enumerating all these candidate points, the one that yields the lowest function value is an optimal solution to the problem. Improved versions of this algorithm are proposed to reduce the number of candidate points that need to be evaluated.

The explicit enumeration approach is still quite expensive for non-trivial problems. Using a systematic enumeration scheme such as B&B to evaluate the partitions is more practical. However, in the x-space, the discontinuities between these partitions are not orthogonal to the (x-) variable axes, thus branching on x-variables would result in discontinuities in the interior of the hyperrectangles defining the nodal problems. To circumvent this difficulty, Ahmed et al. (2004) propose to transform the problem into the space of the tender variable vector χ:

$$\text{TP: } \min\{cx + \sum_s p_s F_s(\chi) | x \in X,\ Tx = \chi, \text{ and (2.30)}\}, \tag{2.33}$$

so that the partitioned hyperrectangles are of the form (2.32b), having constant second-stage function values, and the discontinuities between these hyperrectangles are orthogonal to the (χ-) variable axes. They show that if χ^* is an optimal solution to (2.33), then $x^* \in \text{argmin}\{cx | x \in X,\ Tx = \chi^*\}$ is an optimal solution of (2.1).

Their B&B algorithm can handle general first-stage variables and scenario-dependent recourse. Hence, Assumptions A7 and A8 are not needed. For each hyperrectangular partition $P^k \equiv \prod_{j=1}^{m_2} (l_j^k, u_j^k]$ in the form of (2.31b), since $F_s(\chi)$ is lower semi-continuous and nondecreasing, we can obtain a lower bound on the two-stage problem defined on P^k by solving the following problem:

$$\text{Minimize } cx + \eta \tag{2.34a}$$

$$\text{subject to } x \in X,\ Tx = \chi \tag{2.34b}$$

$$l^k \le \chi \le u^k \tag{2.34c}$$

$$\eta \ge \sum_{s \in S} p_s F_s(l^k + \epsilon), \tag{2.34d}$$

where $F_s(\cdot)$ is as defined in (2.30), and ϵ is calculated *a priori* as a sufficiently small number such that $F_s(\cdot)$ is constant over $(l^k, l^k + \epsilon]$. The value for ϵ is decided as follows. Along each

axis j, $\forall j = 1, \ldots, m_2$, within the unit interval $(k_{1j} - r_{1j} - 1, k_{1j} - r_{1j}]$ for some $k_{1j} \in \mathbb{Z}$, the candidate point of discontinuity for each scenario s is identified as $\lfloor k_{1j} - r_{1j} + r_{sj} \rfloor - r_{sj}$, $\forall s \in S$. These points of discontinuity repeat the same pattern to the right of $k_{1j} - r_{1j}$ with a unit period. It is then sufficient to sort these points, and obtain the smallest interval as ϵ_j along axis j. The final ϵ-value is chosen as strictly smaller than each ϵ_j.

At the node selection step, the partition that yields the least lower bound is selected for further branching. At the branching step, the branching variable $\chi_{j'}$ is selected such that $\chi_{j'} + r_{sj'}$ is an integer greater than $l_{j'}$ and $\mathcal{Q}(\chi)$ is discontinuous at $\chi_{j'}$ for some scenario s. Let $\bar{y}^s$ be the solution obtained from (2.34) for scenario s. For each $j = 1, \ldots, m_2$, let the $p_j \equiv \min_{s \in S}\{(W_s \bar{y}^s)_j - r_{sj}\}$, and let the variable $\chi_{j'}$ be chosen such that $p_{j'}$ lies most in-between its bounds $l^k_{j'}$ and $u^k_{j'}$. Partition P^k is then branched along axis j' at the value $p_{j'}$. Using such a B&B scheme, the algorithm therefore avoids an exhaustive enumeration of the partitions, yet guarantees finite convergence.

Similar to Schultz et al. (1998), Kong et al. (2006) examine integer programs that differ only in their right-hand-sides. In lieu of Gröbner bases, their algorithms use stored value functions of parameterized integer programs. In addition to Assumptions A1-A7, they further assume the following:

A9. The second-stage objective coefficient vector g is fixed.

A10. The first-stage variables are purely integer-restricted, that is, $x \in \mathbb{Z}^{n_1}_+$.

A11. All elements in A, T, b, and r_s, $\forall s \in S$, are integral.

The first and second stage problems now both belong to the type of pure integer programs that have integral coefficient matrices and right-hand-side vectors. In general, given a coefficient matrix $G \in \mathbb{Z}^{m \times n}_+$, this type of integer program can be expressed as:

$$(VF) : z(\zeta) = \min\{dx | Gx \geq \zeta, x \in \mathbb{Z}^n_+\}, \text{ for } \zeta \in \mathbb{Z}^m, \tag{2.35}$$

where $z(\cdot) : \mathbb{Z}^m \mapsto \mathbb{Z}$ is its value function. $z(\cdot)$ is nondecreasing over $\mathbb{Z}^m$ and subadditive over $D = \{\zeta \in \mathbb{Z}^m | \{x \in \mathbb{Z}^n_+ | Gx \geq \zeta\} \neq \emptyset\}$. That is, for $\zeta_1, \zeta_2 \in D$, $\zeta_1 + \zeta_2 \in D$ implies that $z(\zeta_1) + z(\zeta_2) \geq z(\zeta_1 + \zeta_2)$. (The authors use the superadditivity property for maximization problems. We chose to use subadditivity for minimization problems to maintain consistency in exposition.)

Using these properties, Kong et al. (2006) develop an integer programming-based algorithm and a dynamic programming-based algorithm that compute the value of $z(\zeta)$ for each ζ in a finite set Υ. In the integer programming-based algorithm, at each iteration k, some $\zeta^k \in \Upsilon$ is selected, and the corresponding integer program $\min\{dx | Gx \geq \zeta^k, x \in \mathbb{Z}^n_+\}$ is solved to obtain a solution $\hat{x}^k$. Based on the value of $\hat{x}^k$, other $\zeta \in \Upsilon$ are then selected to have their lower bounds, $l(\zeta)$, and upper bounds, $u(\zeta)$, updated using the nondecreasing and subadditivity properties of $z(\zeta)$, until $l(\zeta) = u(\zeta)$, for all $\zeta \in \Upsilon$, at which point, $z(\zeta) = l(\zeta) = u(\zeta)$

are available for all $\zeta \in \Upsilon$. The dynamic programming-based approach does not solve any integer program, but is only applicable when G is nonpositive. Denote Υ_j to be the set of $\zeta \in \Upsilon$ such that $\zeta \leq G_j$. This algorithm uses the initial condition $z(\zeta) = 0$, $\forall \zeta \in \Upsilon \backslash \bigcup_{j=1}^{n} \Upsilon_j$, and the recursive function $z(\zeta) = \min\{d_j + z(\zeta + G_j) : G_j \in \Upsilon, j = 1, \ldots, n\}$, $\forall \zeta \in \bigcup_{j=1}^{n} \Upsilon_j$. Naturally, both algorithms work better when the size of Υ is small.

To use these value function evaluation algorithms to solve two-stage SMIPs, denote the first-stage value function as

$$z_1(\zeta_1) = \min\{cx | Ax \geq b, Tx \geq \zeta_1, x \in \mathbb{Z}_+^{n_1}\},$$

$$\forall \zeta_1 \in \Upsilon^1 \equiv \{\zeta_1 \in \mathbb{Z}^{m_2} | \zeta_1 = Tx \text{ for some } x \in X\},$$

and denote the second-stage value function as

$$z_2(\zeta_2) = \min\{gy | Wy \geq \zeta_2, y \in \mathbb{Z}_+^{n2}\},$$

$$\forall \zeta_2 \in \Upsilon^2 \equiv \bigcup_{\zeta_1 \in \Upsilon^1} \bigcup_{s \in S} \{r_s - \zeta_1\}.$$

These function values for the first and second stage problems are stored for all possible right-hand-side values in Υ^1 and Υ^2, respectively. (Due to the storage requirement, this approach is more suitable for problems having relatively small number of rows.) After storing these value function responses, the feasible region for χ is systematically searched to obtain an optimal value χ^*. A finitely convergent B&B search scheme is proposed. At node k defined on a hyperrectangle $[l^k, u^k]$, a lower bound LB^k and an upper bound UB^k are computed as

$$LB^k = z_1(l^k) + \sum_s p_s z_2(r_s - u^k),$$

and

$$UB^k = z_1(u^k) + \sum_s p_s z_2(r_s - l^k).$$

Branching is performed by partitioning the current hyperrectangle by bisecting some selected axis, which ensures convergence.

To summarize our review on solution approaches for SMIPs, Table 2.1 lists the literature we have covered, grouped by the type of variables appearing in the two stages, which is most relevant to the algorithmic developments. In Chapter 3, we shall propose a decomposition-based B&B algorithm for solving problems having mixed-integer variables in both stages. To our knowledge, the algorithm by Carøe and Schultz (1999) is the only earlier research that is designed for this general class of problems. However, as described in Section 2.1.2.2, their method requires the optimization of several non-smooth Lagrangian dual problems using subgradient methods. Our approach, on the other hand, solves a linear (discrete or continuous) program at each B&B node, and continuous linear programs at the second stage, which is computationally much more attractive.

Table 2.1: Literature on solving stochastic mixed-integer programs.

First stage (x)	Second stage (y)	Literature	Approach	Section
Binary	Continuous	Laporte and Louveaux	L-shaped	2.1.2.1
		Alonso-Ayuso et al.	Branch-and-Fix	2.1.2.2
Continuous	Integer	Schultz et al.	Gröbner basis	2.1.4
Integer	Integer	Hemmecke and Schultz	Test-set decomposition	2.1.2.3
		Kong et al.	Value function	2.1.4
Mixed-integer	Integer	Ahmed et al.	χ-transformation	2.1.4
		Carøe and Tind	Dual pricing	2.1.2.1
Continuous	0-1 mixed-integer	Carøe and Tind	Disj. prog.	2.1.3.1
		Ntaimo and Sen	Disj. prog.	2.1.3.2
Binary	0-1 mixed-integer	Sherali and Fraticelli	Benders & RLT	2.1.3.1
		Sen and Sherali	Disj. prog.	2.1.3.1
		Sen and Higle	Disj. prog.	2.1.3.1
Mixed-integer	Mixed-integer	Carøe and Schultz	Lagrangian dual	2.1.2.2
		Schultz and Tiedemann	Lagrangian dual	2.1.2.2

2.2 Airline Fleet Assignment and Re-fleeting Problems

2.2.1 Fleet Assignment Problem

The fleet assignment problem (FAP) is a major component of the airline scheduling process, and thus, has been studied at considerable length by researchers and practitioners (see Gopalan and Talluri (1998), Yu and Yang (1998), and Sherali et al. (2006), for extensive reviews).

The fleet assignment problem is often formulated as an integer program, mostly as a multi-commodity network flow problem, and a variety of heuristic approaches have been developed to obtain good solutions in reasonable computing times (see, for instance, Abara (1989), Daskin and Panayotopoulos (1989), Hane et al. (1995), Yan and Young (1996), Rushmeier and Kontogiorgis (1997), and the references cited therein). Many of these formulations employ a time-space network, with the earlier ones appear at Levin (1971), Abara (1989), and Hane et al. (1995). We will adopt the type of time-space network by Levin (1971) and Hane et al. (1995) in our stochastic programming fleet assignment model in Chapter 4, and will present the details of such networks therein. Most of these models also include the following three sets of constraints: (a) *Cover Constraints*, which require that each leg be assigned to one aircraft type. (b) *Balance Constraints*, which require that the flow in and out of a node in the time-space network be balanced (i.e., the total number of aircraft flying

in and currently at that station should equal the total number of aircraft that are either flying out or staying at the station). (c) *Availability Constraints* (*Count Constraints*), which require that the number of aircraft of each type assigned to flights cannot exceed the total available number. Hereafter, we will refer to these three sets of constraints as the *basic FAP constraints*. The demand (versus capacity) information, if considered, is usually incorporated via suitable cost terms within the objective function, assuming the independence of *leg* demands, where each leg corresponds to a single airport-to-airport flight.

Several extensions to the FAP have also been studied, such as including aggregate aircraft maintenance considerations (Subramanian et al. (1994) and Clarke et al. (1996)), combining aircraft fleeting and routing (Desaulniers et al. (1997) and Barnhart et al. (1998)), and allowing for flight departure times to vary within a small time-window so that more choices of assigning aircraft to legs are possible (Rexing et al. (2000)). Not surprisingly, the problem size grows considerably with each enhancement. Due to the large sizes of these integer programming models, some researchers have started investigating the effectiveness of alternative approaches for the FAP. For instance, Sosnowska (2000) has presented two heuristics for the FAP, both based on randomized local search procedures.

Most FAP formulations considered in the aforementioned research have an important limitation. They minimize the total fleeting cost (or maximize the revenue) considering an assignment cost for each possible type-and-leg combination in the objective function, based on the corresponding aircraft capacity and the exogenously determined demand on the leg. However, a significant portion of all demand flown by airlines consists of *multiple-leg passengers* (i.e., passengers whose itinerary is comprised of more than one flight leg), especially for major US airlines that utilize hub-and-spoke networks, where passengers fly into and out of hub airports in making flight connections from their origins to their destinations. Clearly, the demand coming from a passenger, who needs to fly multiple legs between a specified origin and destination, will be dependent on the availability of seats on all legs (the *network effect*). Consequently, leg demands can be highly dependent, and passengers who demand to fly on a particular leg are *not identical* in terms of the revenue that they will generate and the airline resources that they will consume.

The efforts to include the network effect in fleet assignment models include Farkas (1995), Kniker (1998), Jacobs et al. (1999), Barnhart et al. (2002), and Sherali et al. (2005). In particular, Kniker (1998) develop a heuristic approach to the fleeting problem, which iterates between the traditional leg-based FAP and a spill model that considers the network effect. Barnhart et al. (2002) include the network effect and the recapture effect in a fleet assignment model formulation, and propose a solution technique that is heuristic in nature due to the large size of the problem. This solution approach is based on a column and row generation technique, along with an LP relaxation based B&B procedure for determining an integer solution. Their computational experiments suggest that incorporating the network and recapture effects can significantly impact the revenue (on the order of $33.7 million to $153.2 million per year for a major US carrier).

2.2.2 Re-fleeting Problem

In this section, we review the research on dynamic re-fleeting as driven by market demand changes. Other types of changes, such as operational disturbances and changes in maintenance requirements, also frequently require fleet assignments to be updated, but in this case, a solution is usually needed in a much shorter time-frame. Thus, this type of re-fleeting focuses on the perturbed constraints, and seeks a one-time correction to the solution at hand, usually via some fast heuristic approach. (See, for instance, Jarrah et al. (1993), Yu and Luo (1997), Thengvall et al. (2000), Rosenberger et al. (2002).) In this section, we will discuss the demand driven re-fleeting issue.

Any re-fleeting decision that is accompanied by changes in the current crew assignments is very difficult and expensive to implement. As a result, it becomes advantageous to confine the re-assignment to lie within the aircraft types having the same cockpit configuration and crew skill requirements, i.e., the same aircraft *family*, and to the legs assigned to this particular family. This way, the re-assignment would only possibly change the aircraft type that was originally assigned to some crew to another type having a different capacity, but yet serviceable by the same crew.

Berge and Hopperstad (1993) are one of the first researchers to study the benefits of *systematically* revising fleet assignments over time. They propose the concept of *Demand Driven Dispatch* (D^3), which is to dynamically change aircraft type assignments as the flight departure times approach and forecasts improve. They explore revisions to assignments that are restricted within each aircraft family, mainly due to the need to preserve crew schedules. For each departure day, they solve a leg-based FAP, using a formulation that is very similar to the leg-based models appearing in the aforementioned literature. Their case studies reveal a potential 1-5% improvement in operating profit from the application of this approach, realized mainly due to spill avoidance and a reduction in the number of larger airplanes used.

From a different perspective, Bish et al. (2004) study the benefits of several Demand Driven Swapping (DDS) mechanisms characterized in terms of their *timing* (when the swapping decision should be made) and *frequency* (how often the swapping decision should be revised). As with any re-fleeting decision, swapping aircraft early in time will cause less disturbance to revenue management and operations, but will be based on more uncertain demand. On the other hand, allowing revisions to the initial swapping decision when demand uncertainty is reduced will benefit from the use of improved demand information, thus reducing the possibility of swaps with loss, but will disrupt operations and revenue management more. In addition, if the swapping decision is delayed too much, then some customers may have already been rejected due to capacity restrictions. The proposed DDS approach is intended to be implemented at close proximity to departures (i.e., six weeks) so as to incorporate the updated demand forecasts into the fleeting. Accordingly, the authors consider swapping aircraft that are assigned to simple loops within similar time-frames where each such loop consists of a round-trip that originates and terminates at a common airport. Using

both analytical models as well as numerical integration and simulation studies, they determine conditions under which each of the different DDS strategies is effective. The proposed methodology is also tested using data obtained from United Airlines. The results reveal that implementing a DDS strategy can significantly improve an airline's revenue.

Pilla et al. (2005) have recently proposed the construction of good approximations to optimal value functions for a two-stage stochastic programming fleet assignment model by employing a design and analysis of computer experiments approach. Similar to the traditional fleet assignment models, the passenger demands and aircraft capacities are incorporated in the cost terms within the objective function, assuming the independence of leg demands. In their experimental design, the design of experiment (DoE) points are a set of first-stage solutions representing aircraft assignment at the family level. In order to obtain a feasible polytope for the first-stage solutions, an aircraft-family assignment problem is constructed using the three basic FAP constraints, aggregated at the aircraft-family level. The equality constraints (cover and balance constraints) and implicit equalities generated from implicit opposing inequalities are used to reduce the dimension of the first-stage solution space. A Latin hypercube algorithm is applied on the resulting first-stage constraint polytope to obtain the set of DoE points. For each DoE point, the second-stage problem that assigns aircraft types to the flight legs in the network is solved for each scenario. The average revenues generated for all DoE points are collected, and a multivariate adaptive regression splines statistical model is used to fit these revenue data to estimate the second-stage function. This fit is expected to be used for future evaluations for the first-stage optimization to obtain a single first-stage decision.

Since the re-fleeting problem is solved for one family at a time, more accurate and detailed path-level (as opposed to leg-based) demand forecasts that are obtained as departures approach can be incorporated into the re-fleeting model. This is the problem studied in Sherali et al. (2005). Although the resulting formulation is somewhat similar to the initial FAP model with path-based demands, this problem is solved on a relatively smaller scale by considering re-fleeting decisions only for the legs that are assigned to a particular family, while keeping the fleet assignments for the other families fixed. The legs that are assigned to the other families, however, still need to be included in the formulation. This is because the fleet types assigned to those legs can still restrict the number of passengers that can be accepted on paths that contain both these legs and those pertaining to the particular family that is to be re-fleeted. They perform a polyhedral analysis of the mixed-integer formulation and design solution approaches using several reformulation and partial convex hull construction mechanisms, along with various classes of valid inequalities to tighten the model representation. Properly composing these strategies greatly enhances the overall effectiveness of the model, inducing its LP relaxation to yield a relatively greater number of variables that automatically turn out to be binary-valued. This polyhedral analysis based approach was also applied to solve a real problem posed by United Airlines, having $5,098$ flight legs (with 459 legs in the family) and $64,247$ paths, within 20 cpu minutes of computing time on a Sun-Blade-1000 (UltraSPARC-III) workstation having 512 MB RAM and

a cpu speed of 750 MHz. The tight relaxations also facilitated the design of an effective heuristic for solving relatively larger sized problems. The results indicated that the solution effort required was significantly reduced, while the quality of the solution was only mildly degraded.

The foregoing study will be used in Chapter 4, which is a significant extension of this work that incorporates stochastic demands in the initial fleet assignment phase for increased flexibility. We shall propose a two-stage stochastic model that makes assignment decisions for multiple aircraft families in the first stage, and incorporates future re-fleeting opportunities in the second stage. Uncertain path-level demands are reflected by scenarios in second-stage re-fleeting subproblems. These render the size of this problem incomparable by its single-family, deterministic predecessor.

2.3 Workforce Planning Models

Research on personnel arrangement to meet workforce demand can be found in areas such as workforce scheduling and workforce planning. Workforce scheduling problems generally deal with determining work schedule patterns and assigning workers to jobs. Such problems include the considerations of off-days and work-stretches (consecutive work days), cyclic schedules, multiple shifts, varying (cyclic) demand, time-windows and teaming-constraints, hierarchical workforce, and their various combinations. Compared with workforce scheduling, workforce planning deals with higher level and longer term workforce needs in terms of availability, allocation, and transition. There are two types of workforce planning models addressed in the literature (Price et al. (1980), Purkiss (1981)): descriptive (or exploratory) models and normative models. Descriptive models are analytical tools that *predict* how the workforce system would respond to given scenarios. Examples include, but are not limited to, Markov (cross-sectional) models, renewal models, semi-Markov models, and simulation models. Normative models are optimization tools that *prescribe* policies so as to achieve an ideal workforce system under given criteria. In the following, we selectively review some models from each of these two types.

2.3.1 Descriptive Workforce Planning Models

Edwards and Morgan (1982) provide a systematic review on some early descriptive models, and Edwards (1983) summarizes the application packages based on these models in the U.K. with emphasis on data requirement and representation. McClean (1991) reports some Markov chain models and their input data estimations, and Ugwuowo and McClean (2000) focus on workforce wastage modeling in their review paper. In this subsection, we follow the terminology and notation in Bartholomew and Forbes (1979) to discuss some basic ideas of descriptive workforce planning models.

Many workforce systems can be classified into categories according to, for instance, hierarchies, grades, or skill sets. The *stocks* refer to the workforce in such category at a given time, and the *flows* refer to the changes between these categories over a period of time. There are two extremes of flow types: the "push" flows and the "pull" flows. In the former, the system flows are "pushed" from the sources, such as the recruitment of new members pushing the flow from the lower end of the hierarchy; in the latter, the flows are "pulled" from the end, for example, retirement leaving vacancies for younger members to fill in. The push system is mostly represented using Markov models, and the pull system is generally interpreted by renewal models.

Markov Models

Markov models are also called cross-sectional models (Grinold and Marshall (1977)). The basic form is

$$n(t) = n(t-1)P + R(t)r, \tag{2.36}$$

where $n(t)$ is the (row) vector of stocks at time t, P is the transition matrix between the categories, $R(t)$ is the number of recruits at time t, and r is the vector denoting the probabilities of a new recruit being assigned to each stock category.

If the ideal workforce size of the entire system is given, then the number of recruits is a function of the vacancies in the system. Define $N(t)$ as the size of the system at time t, and w as the *wastage* vector to indicate the rate at which members leave the system. The number of recruits is thus

$$R(t) = N(t) - N(t-1) + n(t-1)w^T, \tag{2.37}$$

and (2.36) becomes the so-called *fixed-size model*:

$$n(t) = n(t-1)Q + M(t)r, \tag{2.38}$$

where $Q = P + w^T r$ and $M(t) = N(t) - N(t-1)$.

The Markov models (2.36) and (2.38) assume proportions of flows to be given probabilities. Morgan (1979) includes decision variables for recruitment and promotion rates into his hierarchical-system model. As we shall see later, many normative models use Markov models as starting points, or constraints, for prescribing optimal solutions. In models (2.36) and (2.38), each stock size is free to vary as a result of workforce flows. However, in reality, stock size changes are not without constraints. System flows such as recruitment and promotion are often "pulled" by vacancies in upper-levels of the hierarchy as they appear. This "fill-in" idea is used by renewal models.

Renewal Models

Renewal models consider the times at which successive job occupancies appear as constituting renewal points of the process. Consider a simple system having multiple, say, K, levels of a hierarchy and fixed size at each level. Whenever a vacancy occurs at level k, it

must be filled, either through recruitment or promotion from the immediate lower level. Let the fixed level sizes be n_k and the wastage rates be w_k, and denote the promotion probabilities as s_k and the recruitment probabilities as $1 - s_k$, $k = 1, \ldots, K$. Letting $P_{k,k+1}$ be the expected number of promotions between levels k and $k+1$ over one time interval, we have, for the highest level K:

$$P_{K-1,K} = n_K w_K s_K. \tag{2.39a}$$

At lower levels, we have,

$$\begin{aligned} P_{k,k+1} &= (\text{number of vacancies in } k+1) s_{k+1} && (2.39b)\\ &= (n_{k+1} w_{k+1} + P_{k+1,k+2}) s_{k+1}, \quad \forall k = 1, \ldots, K-2. && (2.39c) \end{aligned}$$

Therefore, in general,

$$P_{k,k+1} = \sum_{i=k+1}^{K} n_i w_i \prod_{h=k+1}^{i} s_h, \quad \forall k = 1, \ldots, K-1, \tag{2.39d}$$

and the recruitment numbers R_k can be obtained via

$$R_k = n_k w_k + P_{k,k+1} - P_{k-1,k}, \forall k = 1, \ldots, K, \tag{2.39e}$$

where $P_{0,1} = P_{K,K+1} = 0$.

More variations and applications of the renewal models are discussed in, for example, Bartholomew and Forbes (1979), Bartholomew (1982), Yadavalli and Natarajan (2001). Compared with Markov models, renewal models, along with semi-Markov models, are computationally more challenging; yet, these models are sometimes more realistic and they provide powerful tools to describe workforce systems.

2.3.2 Normative Workforce Planning Models

The descriptive models are mostly used in imitating the system, and can help decision-makers predict the status of the system and test various policies. Normative models, mostly mathematical programs, prescribe policies according to system constraints and management criterion.

Dynamic Programs

If the objective of the workforce system is to minimize the total costs of recruitment, retention, over- and under-staffing, as well as firing/retirement, a dynamic programming formulation can be constructed similar to the well-known Wagner-Whitin model (Wagner and Whitin (1952)). (Also, see Rao (1990) for such a construction). Mehlmann (1980) applies dynamic programming in a Markov model to determine optimal recruitment and transition patterns. The recursive objective function is a quadratic function that penalizes discrepancies between the actual and the preferred patterns.

Goal Programs

In workforce planning, goal programming is used when multiple objectives are to be met with a certain degree of satisfaction, such as desired workforce size, personnel flow, and related costs. Price and Piskor (1972) present a goal programming application to the Canadian Armed Forces with ideal promotion rate and force strength as goals. Charnes et al. (1976) use goal programming with an embedded Markov process to model the U.S. Navy system to address the issue of "Equal Employment Opportunity". Georgiou and Tsantas (2002) combine Markov model and goal programming to model a workforce system, in which new members need to go through a training period, and formal members are "recruited" from trainees. The goal program constraints are from the workforce system flow study using a fixed-size Markov model. The "goals" are the desired workforce level and training costs, and the objective is to lexicographically, for ordered priority groups, minimize the weighted sum of the deviations from these goals. Corominas et al. (2005) formulate a goal program to assign multi-category workers with minimum and desired functional workforce size. They use nonlinear terms to penalize personnel shortages and surpluses. The variable integrality renders the objective function as piecewise-convex, and the problem is transformed and solved as a convex cost network flow problem.

Stochastic Programs

Variations in service levels and the uncertainty of workforce requirement lead to several chance-constrained stochastic programming workforce planning models. Abernathy et al. (1973) apply an iterative approach to decide workforce policy, staff planning, and short-term scheduling. Given a workforce policy decision, the inner loop solves staff planning and scheduling using a stochastic program, and the result is sent to the outer loop for further policy decisions. The stochastic model used in the inner loop considers workforce demand as a random variable. The chance-constraint specifies an upper bound (risk level) on the probability that workforce size turns out to be less than the demand. A Monte Carlo model is used to process the chance-constraint and determine a lower bound for the workforce size. Charnes et al. (1974) also sketch a chance-constrained model in which the probabilities of the budget and the workforce size satisfying certain requirements are stipulated to be greater than some specified levels. The inverses of the marginal distribution functions need to be identified for the model to be solvable as a usual (linear) goal program.

Aside from chance-constrained models, El Agizy (1971) includes stochasticity in the objective function, using a term to represent the expected workforce costs due to uncertain workload. The stochastic cost term is estimated using a piecewise linear function. Martel and Price (1977) consider a multi-period stochastic goal programming model with simple recourse, where workforce surpluses and shortages are accommodated using the recourse variables. The budget and demand are assumed to follow Beta distributions. The resulting convex (but nonlinear) terms in the objective function are approximated using piecewise linear functions.

In Chapter 5, we shall use a two-stage stochastic program with recourse to model a multi-

category workforce planning situation, in which workforce demand is uncertain, yet workload can be adjusted after employees are hired and allocated. Different from the models by El Agizy (1971) and Martel and Price (1977), the uncertainties in demand shall be considered in constraint coefficients, resulting in a model having a stochastic recourse matrix.

Chapter 3

On Solving Discrete Two-Stage Stochastic Programs Having Mixed-Integer First- and Second-Stage Variables

3.1 Introduction

As the applications of stochastic mixed-integer programs (SMIP) emerge in increasingly wider areas, the need for efficient algorithms that are practical in solving real-sized (usually very-large-scale) problems of this type becomes pressing. As mentioned in Chapter 2, the existing research thus far has been limited to two-stage SMIPs having purely integer or continuous variables in either stages. We significantly enhance this arena by studying a relatively general class of SMIPs that includes mixed-integer variables in both stages. For this class of problems, we develop a decomposition-based branch-and-bound algorithm in this chapter, based on certain key theoretical results and novel algorithmic components. This paves the way for the following two chapters, in which we study two real-life applications of SMIPs, and develop tailored effective solution procedures.

Consider then, the following wide class of mixed-integer two-stage stochastic programs:

$$\textbf{SMIP:} \quad \text{Minimize} \;\; c^T x + \sum_{s \in S} p_s f_s(x) \tag{3.1a}$$

$$\text{subject to } x \in X \cap \Omega \tag{3.1b}$$

where

$$X \equiv \{x \in R^n : \; Ax \geq b, \; x_i \geq 0, \; \forall i \in I_1 \subseteq \{1, \ldots, n\}, \; x_i \text{ binary}, \; \forall i \in I_2 \equiv \{1, ..., n\} \setminus I_1\} \tag{3.1c}$$

$$\Omega \equiv \{x \in R^n : \; l \leq x \leq u\} \text{ (with } l_i \equiv 0 \text{ and } u_i \equiv 1, \; \forall i \in I_2) \tag{3.1d}$$

and for any scenario $s \in S$, we have,

$$f_s(x) = \quad \text{minimum} \quad g^T y \tag{3.1e}$$

$$\text{subject to} \quad W_s y \geq r_s - T_s x \tag{3.1f}$$

$$y \in Y \equiv \{y \in R^m : y_j \geq 0, \; \forall j \in J_1 \subseteq \{1, \ldots, m\}, \; y_j \text{ binary}, \; \forall j \in J_2 \equiv \{1, ..., m\} \setminus J_1\}. \tag{3.1g}$$

We make the following assumptions:

A1. The continuous variables in both stages are bounded. Moreover, the continuous variables in the second stage are scaled onto [0,1], with the corresponding bounding restrictions $y_j \leq 1$, $\forall j \in J_1$, being absorbed within (or implied by) (3.1f).

A2. The inherent stochasticity in the problem is discretized into so-called scenarios, $s \in S$, each with an associated probability of occurrence p_s, $\forall s \in S$.

A3. For any $x \in \Omega$, the region defined by (3.1f, 3.1g) is feasible (relative complete recourse with respect to Ω).

The remainder of this chapter is organized as follows. Section 3.2 introduces several important concepts that lay the foundation for designing a decomposition-based branch-and-bound approach for solving two-stage stochastic programs of the form given in SMIP (3.1). The main algorithmic procedure is described in this section and its convergence to a global optimum is established. Section 3.3 then presents the supporting algorithmic routines for solving the related subproblems and the lower-bounding master programs. An illustrative example is provided in Section 3.4, and Section 3.5 reports some encouraging preliminary computational experience. Finally, Section 3.6 concludes this chapter.

3.2 Decomposition-Based Branch-and-Bound Algorithm (DBAB)

In this section, we begin by presenting some fundamental concepts and results that lead us to design our proposed overall algorithm for solving Problem SMIP. Observe that when the first-stage variables x are purely binary, then for each fixed binary vector $\bar{x}$, the extreme points of $conv\{(x,y) : T_s x + W_s y \geq r_s, x \in \Omega, y \in Y\} \cap \{(x,y) : x = \bar{x}\}$ have binary values for y_j, $\forall j \in J_2$, for each scenario $s \in S$. This follows because the restriction $x = \bar{x}$ is then facial with respect to Ω. However, this statement is no longer true when the first stage contains continuous variables, whereby, x might be fixed at some nonextremal point within Ω. As a result, the algorithms developed by Sen and Higle (2005), Sherali and Fraticelli (2002), and Sen and Sherali (2006), in particular, all of which rely on this basic construct, are no longer applicable in this case. This necessitates the development of an alternative solution approach.

Toward this end, for any fixed $x \in \Omega$, consider the scenario-based value functions

$$f_s(x) \equiv \min\{g^T y : W_s y \geq r_s - T_s x,\ y \in Y\},\ \forall s. \tag{3.2}$$

Now, given any Ω (which will be partitioned in a B&B context in the sequel), define

$$Z_s(\Omega) \equiv conv(\Gamma_s(\Omega)),\ \text{where} \tag{3.3a}$$

$$\Gamma_s(\Omega) \equiv \{(x,y) : T_s x + W_s y \geq r_s, x \in \Omega, y \in Y\},\ \forall s. \tag{3.3b}$$

By a convexification process (e.g., see Sherali and Adams (1999)), suppose that we have the representation of $Z_s(\Omega)$ in a possibly higher dimensional space (x,y,z), including certain new variables z in the added dimensions, given as follows:

$$Z_s(\Omega) = \{(x,y) : H_{1s}x + H_{2s}y + H_{3s}z \geq h_s, x \in \Omega, (y,z) \geq 0\},\ \forall s. \tag{3.3c}$$

Based on this representation, define the following function for any fixed $x \in \Omega$:

$$LB_s(x) \equiv \min\{g^T y : H_{2s}y + H_{3s}z \geq h_s - H_{1s}x, (y,z) \geq 0\} = \underset{y:(x,y)\in Z_s(\Omega)}{\text{minimum}} \{g^T y\},\ \forall s. \tag{3.4}$$

Proposition 3.2.1. For any $s \in S$, consider the functions $f_s(x)$ and $LB_s(x)$ as defined by (3.2) and (3.4), respectively, over $x \in \Omega$. Then, we have,

$$f_s(x) \geq LB_s(x), \forall x \in \Omega. \tag{3.5a}$$

Furthermore, if $x \in vert(\Omega)$, where $vert(\Omega)$ denote the vertices of Ω, or, more generally, if there exists an optimal solution $\bar{y}$ that evaluates $LB_s(x)$ such that $\bar{y} \in Y$, then we have

$$f_s(x) = LB_s(x). \tag{3.5b}$$

Proof: Consider any fixed $\bar{x} \in \Omega$. Define

$$Z_{LB_s}(\bar{x}) \equiv \{y : (\bar{x}, y) \in Z_s(\Omega)\}, \text{ and } Z_{f_s}(\bar{x}) \equiv conv\{y : (\bar{x}, y) \in \Gamma_s(\Omega)\}. \tag{3.6a}$$

Hence, from (3.2), (3.3b), and (3.4), we have,

$$f_s(\bar{x}) = \min\{g^T y : y \in Z_{f_s}(\bar{x})\} \text{ and } LB_s(\bar{x}) = \min\{g^T y : y \in Z_{LB_s}(\bar{x})\}. \tag{3.6b}$$

Note that $Z_{LB_s}(\bar{x})$ is given by $conv(\Gamma_s(\Omega)) \cap \{(x, y) : x = \bar{x}\}$, while $Z_{f_s}(\bar{x})$ is given by $conv(\Gamma_s(\Omega) \cap \{(x, y) : x = \bar{x}\})$. Hence,

$$Z_{f_s}(\bar{x}) \subseteq Z_{LB_s}(\bar{x}), \forall \bar{x} \in \Omega, \text{ with } Z_{f_s}(\bar{x}) = Z_{LB_s}(\bar{x}), \forall \bar{x} \in vert(\Omega), \tag{3.7}$$

where the latter statement holds true because $x = \bar{x}$ is facial with respect to $conv(\Gamma_s(\Omega))$ for $\bar{x} \in vert(\Omega)$. The results (3.5a) and (3.5b) now follow from (3.6b) and (3.7) and the fact that if an optimal solution $\bar{y}$ that evaluates $LB_s(\bar{x})$ is also feasible to $Z_{f_s}(\bar{x})$, then $f_s(\bar{x}) = LB_s(\bar{x})$. This completes the proof. □

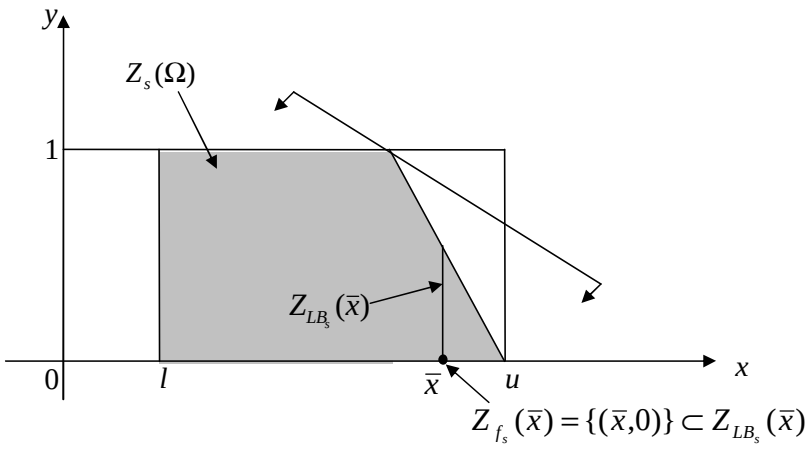

Figure 3.1: Illustration of the concepts of Proposition 3.2.1.

Figure 3.1 illustrates the concepts of Proposition 3.2.1 for a single binary variable y and a continuous variable x, by displaying a situation where for $\bar{x} \notin vert(\Omega)$, we have $Z_{f_s}(\bar{x}) \subset Z_{LB_s}(\bar{x})$, whereas for either $\bar{x} = l$ or $\bar{x} = u$, it is clear that $Z_{f_s}(\bar{x}) = Z_{LB_s}(\bar{x})$.

Consequently, based on Proposition 3.2.1 and (3.4), we get for any $x \in \Omega$,

$$f_s(x) \geq LB_s(x) = \max\{\phi(h_s - H_{1s}x) : \ \phi H_{2s} \leq g^T, \ \phi H_{3s} \leq 0, \ \phi \geq 0\}, \ \forall s. \tag{3.8}$$

Define

$$\Lambda_s \equiv \{\phi : \ \phi H_{2s} \leq g^T, \ \phi H_{3s} \leq 0, \phi \geq 0\}, \quad \forall s \in S. \tag{3.9}$$

Then, from (3.8) and (3.9), we get for any $x \in \Omega$ (since $LB_s(x)$ is finite by assumptions A1 and A3)

$$f_s(x) \geq \max_{\phi^i \in vert(\Lambda_s)} \{\phi^i(h_s - H_{1s}x)\}, \quad \forall s \in S. \tag{3.10}$$

Accordingly, let us define the following *lower bounding master program* (LBMP(Ω)), which is designated based on a specification of Ω for subsequent use.

$$\begin{aligned}
\textbf{LBMP}(\boldsymbol{\Omega}): \quad \text{Minimize} \quad & c^T x + \sum_s p_s \eta_s && \text{(3.11a)}\\
\text{subject to} \quad & \eta_s \geq \phi^i(h_s - H_{1s}x), \quad \forall \phi^i \in vert(\Lambda_s),\ \forall s \in S && \text{(3.11b)}\\
& x \in X \cap \Omega. && \text{(3.11c)}
\end{aligned}$$

Consider the following result.

Proposition 3.2.2. Problem LBMP(Ω) provides a lower bound for SMIP. Moreover, if $(\bar{x}, \bar{\eta}_s, s \in S)$ solves LBMP(Ω), and either (a) $\bar{x} \in vert(\Omega)$, or more generally, (b) the solution $\bar{y}_s$ obtained when solving $LB_s(\bar{x})$ satisfies $\bar{y}_s \in Y$, $\forall s \in S$, then $\bar{x}$ solves SMIP with the same objective value given by $c^T\bar{x} + \sum_s p_s\bar{\eta}_s$.

Proof: Note that from (3.1), (3.10), and (3.11), we have that LBMP(Ω) provides a lower bound on SMIP. Moreover, by Proposition 3.2.1, we have that under either of the conditions (a) or (b) of the proposition, equality holds true in (3.5b) or (3.10), and so, $\bar{x}$ solves SMIP with the same objective value as for LBMP(Ω). This completes the proof. □

Proposition 3.2.2 leads to a decomposition-based B&B algorithm for solving SMIP, which is predicated on the partitioning of the hyperrectangle Ω. In this procedure (a more formal statement is given below), starting with iteration $k = 1$ and a list of active nodes $L_1 \equiv \{1\}$, where $\Omega^1 \equiv \Omega$, at any general iteration k, we will have a current index list L_k of active (non-fathomed) nodes, where each $\lambda \in L_k$ corresponds to some hyperrectangle $\Omega^\lambda \equiv \{x : l^\lambda \leq x \leq u^\lambda\} \subseteq \Omega$. For each such node, we will have computed a lower bound LB_λ via the solution of LBMP(Ω^λ). (The algorithm described below, as well as its convergence arguments, remains the same if the lower bounds are computed by instead solving $\overline{\text{LBMP}}(\Omega^\lambda)$, which is given by (3.11) but with the integrality restrictions on x_i, for $i \in I_2$, relaxed. Naturally, such a lower bound is potentially weaker, although relatively easier to compute.) Note that LBMP(Ω^λ) (or $\overline{\text{LBMP}}(\Omega^\lambda)$) will be solved via a standard row-generation or Benders' scheme, by iterating between the master program (3.11) and the subproblems (3.4) or (3.8), for each $s \in S$, with $\Omega \equiv \Omega^\lambda$ (see Section 3.3.2 for details). Whenever the lower bounding solution x^λ, say, for any node subproblem λ turns out to satisfy either of the conditions of Proposition 3.2.2 with respect to Ω^λ, then the corresponding value of LBMP(Ω^λ) provides the actual value of SMIP restricted to Ω^λ, and we can fathom this node, updating the incumbent solution and its value, x^* and ν^*, respectively, as possible. Additionally, if $LB_\lambda \geq \nu^*$, we can fathom node λ. Note that during the process of computing LB_λ via the solution of the relaxed master programs (3.11), whenever the current master program value exceeds ν^*, we can fathom the

node (without actually computing LB_λ exactly). Therefore, at any iteration k, all active nodes $\lambda \in L_k$ will satisfy $LB_\lambda < \nu^*$, and we will select

$$\lambda(k) \in \text{argmin}\{LB_\lambda : \ \lambda \in L_k\} \tag{3.12}$$

and proceed by partitioning the corresponding hyperrectangle $\Omega^{\lambda(k)}$ into two sub-hyperrectangles based on a *branching variable* x_p selected according to the following rule, where $x^{\lambda(k)}$ is the optimal solution obtained for LBMP($\Omega^{\lambda(k)}$).

Branching Rule A:
Define

$$\theta_i \equiv \min\{x_i^{\lambda(k)} - l_i^{\lambda(k)}, \ u_i^{\lambda(k)} - x_i^{\lambda(k)}\}, \quad \forall i = 1, \ldots, n, \tag{3.13a}$$

and select

$$p \in \underset{i=1,\ldots,n}{\text{argmax}}\{\theta_i\}. \tag{3.13b}$$

Accordingly, partition $\Omega^{\lambda(k)}$ into two sub-hyperrectangles by partitioning the interval $[l_p^{\lambda(k)}, u_p^{\lambda(k)}]$ into two sub-intervals as follows

$$\left.\begin{array}{c} [l_p^{\lambda(k)}, x_p^{\lambda(k)}] \text{ and } [x_p^{\lambda(k)}, u_p^{\lambda(k)}], \text{ if } p \in I_1; \\ [0,0] \text{ and } [1,1], \text{ if } p \in I_2. \end{array}\right\} \tag{3.13c}$$

(Note that the case $p \in I_2$ arises only when $\overline{\text{LBMP}}(\Omega^\lambda)$ is used to compute lower bounds.) The resulting decomposition based branch-and-bound (**DBAB**) algorithm for solving SMIP is stated formally below. (As mentioned above, LBMP($\cdot$) may be replaced by $\overline{\text{LBMP}}(\cdot)$ throughout the following discussion in this section, with the added premise that the updating of incumbent solutions via Proposition 3.2.2 should additionally verify that $\bar{x}$ satisfies the integrality restrictions defining X.)

Algorithm DBAB:

Step 0: Initialization Step. Initialize the incumbent solution x^* to be null and set its objective value as $\nu^* = \infty$. Let the iteration counter $k = 1$, the number of nodes enumerated $N = 1$, and commence with the list of active nodes $L_k = \{1\}$, and set $\Omega^1 = \Omega$, and $\lambda(k) = 1$, with $[l^1, u^1] \equiv [l, u]$. Use the prescribed decomposition algorithm to solve LBMP(Ω^1), and let x^1 be the solution obtained of objective value $LB_1 = \nu[\text{LBMP}(\Omega^1)]$. If either of the conditions of Proposition 3.2.2 holds true, then stop with $x^* = x^1$ as an optimal solution to SMIP having an objective value $\nu^* = LB_1$. Otherwise, select a branching variable x_p via (3.13a) and (3.13b), let $\epsilon \geq 0$ be a chosen optimality tolerance, and proceed to Step 1.

Step 1: Partitioning Step. Partition $\Omega^{\lambda(k)}$ for the selected active node $\lambda(k)$ into two sub-hyperrectangles according to (3.13c) based on the identified branching variable x_p. Replace $\lambda(k)$ within L_k by these two new node indices, $N + 1$ and $N + 2$.

Step 2: Bounding Step. Solve the problem LBMP(Ω^{N+t}) for $t = 1, 2$, corresponding to each of the two new nodes generated. If x^{N+t} satisfies either of the conditions of Proposition 3.2.2 with respect to Ω^{N+t} for any of $t = 1, 2$, then update the incumbent solution x^* and its objective value ν^*, if necessary.

Step 3: Fathoming Step. Fathom any non-improving nodes by setting $L_{k+1} = L_k \setminus \{\lambda \in L_k : LB_\lambda + \epsilon \geq \nu^*\}$. If $L_{k+1} = \emptyset$, then stop with the incumbent solution as ϵ-optimal. Otherwise, select and store branching variable indices using (3.13a) and (3.13b) for each of the two new nodes generated if still active (i.e., belong to L_{k+1}), increment k by one, N by two, and proceed to Step 4.

Step 4: Node Selection Step. Select an active node $\lambda(k) \in \text{argmin}\{LB_\lambda : \lambda \in L_k\}$ as in (3.12), and return to Step 1. □

Proposition 3.2.3. (**Convergence result**). Algorithm DBAB (run with $\epsilon \equiv 0$) either terminates finitely with the incumbent solution being optimal to Problem SMIP, or else, we get $k \to \infty$ such that along any infinite branch of the B&B tree that is associated with the nested sequence of partitions $\{\Omega^{\lambda(k)}\}$, $k \in K_1$, say, any accumulation point of the corresponding solution sequence $\{x^{\lambda(k)}\}_{K_1}$ solves SMIP.

Proof: The case of finite termination is clear from the derivation of the algorithm. Hence, suppose that $k \to \infty$, and consider any infinite branch of the B&B tree generated as identified by the proposition. Over some convergent subsequence indexed by $K_2 \subseteq K_1$, if necessary (noting the boundedness of the sequence generated), let

$$\{x^{\lambda(k)}, l^{\lambda(k)}, u^{\lambda(k)}\}_{K_2} \to (x^*, l^*, u^*). \tag{3.14}$$

We need to show that x^* solves Problem SMIP.

First of all, note that since $LB_{\lambda(k)}$ is the least lower bound among all active nodes at each iteration k, we have that

$$\nu[\text{SMIP}] \geq LB_{\lambda(k)}, \quad \forall k \in K_2. \tag{3.15}$$

Next, note that we can equivalently view LBMP($\Omega^{\lambda(k)}$) as follows:

$$\text{Minimize } \{c^T x + \sum_s p_s g^T y_s : (x, y_s) \in Z_s(\Omega^{\lambda(k)}),\ \forall s,\ x \in X \cap \Omega^{\lambda(k)}\},$$

where $(x^{\lambda(k)}, y_s^{\lambda(k)}, \forall s)$ solves LBMP($\Omega^{\lambda(k)}$). By (3.14) and the boundedness of $Z_s(\cdot)$, $\forall s$, and by replacing K_2 by an appropriate subsequence, if necessary, suppose that $\{(x^{\lambda(k)}, y_s^{\lambda(k)}, \forall s)\}_{K_2} \to (x^*, y_s^*, \forall s)$. Since $(x^*, y_s^*, \forall s)$ is feasible to LBMP(Ω^*), where $\Omega^* \equiv \{x : l^* \leq x \leq u^*\}$, we have that

$$c^T x^* + \sum_s p_s g^T y_s^* \geq \nu[\text{LBMP}(\Omega^*)]. \tag{3.16}$$

We now show that equality must hold true in (3.16). Suppose on the contrary that

$$c^T x^* + \sum_s p_s g^T y_s^* > c^T \hat{x} + \sum_s p_s g^T \hat{y}_s = \nu[\text{LBMP}(\Omega^*)] \tag{3.17}$$

where $(\hat{x}, \hat{y}_s, \forall s)$ solves LBMP(Ω^*). Because $\{\Omega^{\lambda(k)}\}_{K_2}$ is a nested sequence, we have that $\Omega^* \subseteq \Omega^{\lambda(k)}$, $\forall k \in K_2$, and so $Z_s(\Omega^*) \subseteq Z_s(\Omega^{\lambda(k)})$, $\forall k \in K_2$. Consequently, $(\hat{x}, \hat{y}_s, \forall s)$ is feasible to LBMP($\Omega^{\lambda(k)}$), $\forall k \in K_2$, and since $\{c^T x^{\lambda(k)} + \sum_s p_s g^T y_s^{\lambda(k)}\}_{K_2} \rightarrow c^T x^* + \sum_s p_s g^T y_s^*$, we have that for $k \in K_2$ large enough,

$$c^T x^{\lambda(k)} + \sum_s p_s g^T y_s^{\lambda(k)} > c^T \hat{x} + \sum_s p_s g^T \hat{y}_s,$$

which contradicts the optimality of $(x^{\lambda(k)}, y_s^{\lambda(k)}, \forall s)$ for LBMP($\Omega^{\lambda(k)}$). Hence, equality holds true in (3.16), and so, viewing LBMP(Ω^*) in the projected x-space, we have that

$$\{\Omega^{\lambda(k)}\}_{K_2} \rightarrow \Omega^* \equiv \{x : l^* \leq x \leq u^*\}, \text{ where } x^* \text{ solves LBMP}(\Omega^*). \tag{3.18}$$

Now, in the infinite sequence of iterations indexed by $k \in K_2$, there exists some variable x_p, $p \in I_1$, that is branched infinitely often according to (3.13a)–(3.13c). Let $K_3 \subseteq K_2$ correspond to iterations at which x_p is selected as the branching variable. By virtue of the partitioning scheme (3.13c), we know that in the nested sequence of intervals for x_p generated over $k \in K_3$, we have that $x_p^{\lambda(k)} \notin (l_p^{\lambda(k')}, u_p^{\lambda(k')})$, $\forall k' \in K_3$, $k' > k$, while $x_p^* \in [l_p^*, u_p^*]$. Hence, we must have $x_p^* = l_p^*$ or $x_p^* = u_p^*$. By (3.13a), this means that $\theta_p \rightarrow 0$, which in turn implies from (3.13b) that

$$\theta_i \rightarrow 0, \quad \forall i = 1, \ldots, n, \text{ and so, } x^* \in vert(\Omega^*). \tag{3.19}$$

By (3.18) and Proposition 3.2.2, we therefore have that the limiting solution x^* and its value LBMP(Ω^*) provide an upper bounding solution and value, respectively, for Problem SMIP. Hence,

$$c^T x^* + E[f(x^*, \tilde{\omega})] = \nu[\text{LBMP}(\Omega^*)] = \lim_{\substack{k \rightarrow \infty \\ k \in K_3}} LB_{\lambda(k)} \geq \nu[\text{SMIP}]. \tag{3.20a}$$

But since $\{LB_{\lambda(k)}\}_{K_2}$ is monotone increasing and bounded from above, and noting that $K_3 \subseteq K_2$, we get from (3.15) that

$$\nu[\text{SMIP}] \geq \lim_{\substack{k \rightarrow \infty \\ k \in K_3}} LB_{\lambda(k)}. \tag{3.20b}$$

Hence, from (3.20a) and (3.20b), x^* solves SMIP, and this completes the proof. □

Proposition 3.2.4. (**Alternative Branching Rules**). The result of Proposition 3.2.3 continues to hold true under the following two alternative branching rules:
Branching Rule B. Define θ_i, $\forall i$, as in (3.13a), select p as in (3.13b), but replace (3.13c) in Branching Rule A by

$$\left.\begin{array}{l} [l_p^{\lambda(k)}, \frac{l_p^{\lambda(k)} + u_p^{\lambda(k)}}{2}], [\frac{l_p^{\lambda(k)} + u_p^{\lambda(k)}}{2}, u_p^{\lambda(k)}], \text{ if } p \in I_1; \\ [0,0] \text{ and } [1,1], \text{ if } p \in I_2. \end{array}\right\} \tag{3.21}$$

Branching Rule C. Select p according to

$$p \in \underset{i=1,\ldots,n}{\text{arglexmax}}\{(u_i^{\lambda(k)} - l_i^{\lambda(k)}, \theta_i)\} \tag{3.22}$$

where θ_i is defined by (3.13a), and then replace (3.13c) in Branching Rule A by (3.21).

Proof: Following the proof of Proposition 3.2.3, Branching Rule B yields that $x_p^* = l_p^* = u_p^*$ by virtue of the bisection process for indices in I_1 as per (3.21). Hence, $\theta_p \to 0$ in the proof of Proposition 3.2.3, which then implies that (3.19) holds true. The remainder of the proof proceeds identically. Likewise, for Branching Rule C, since (3.22) selects an index having the largest interval (with the first priority), and adopts the partitioning scheme of (3.21), we get that $l^* = u^*$ in this case, which again leads to (3.19) holding true, with the remainder of the proof following that of Proposition 3.2.3. This completes the proof. □

Remark 3.2.1. Note that when $|J_2|$ is relatively small, the entire set (3.3c) can be generated *a priori* and then used to solve the subproblems (3.4) that evaluate $LB_s(x)$, $\forall s$, in a decomposition approach for solving LBMP(Ω). Alternatively, when it is not computationally viable to *a priori* generate the entire convex hull representation as in (3.3c), the subproblems given by (3.4) can be solved via a sequential convexification procedure that generates cutting planes as necessary, which are valid for $Z_s(\Omega)$ of (3.3a). The corresponding master program constraint (3.11b) can then be generated from the resulting linear program at the termination of this scheme. The details for such a finite cutting plane procedure for solving Problem (3.4) are given in Section 3.3.1 below. □

3.3 Algorithmic Routines for Solving Subproblems and Master Programs

In this section, we present algorithmic procedures for solving the subproblem (3.4) for any given $\bar{x} \in X \cap \Omega$ (**Algorithm SP**), and for solving the lower-bounding master program LBMP(Ω) (or its relaxation $\overline{\text{LBMP}}(\Omega)$) as given by (3.11) for any Ω (**Algorithm LBMP**). These routines are described in turn below.

3.3.1 Cutting Plane Procedure for Solving Subproblems of Equation (3.4)

In essence, we can follow the cutting plane scheme described in Sherali and Fraticelli (2002) to sequentially construct valid inequalities for $Z_s(\Omega)$ of (3.3a) in order to solve the subproblem (3.4). However, the difficulty in directly implementing this scheme lies in the fact that when x is fixed at $\bar{x}$, we may not always have y_j binary for all $j \in J_2$ at the extreme points

of $Z_{LB_s}(\bar{x})$ as defined in (3.6a). We thus need to be able to detect whether Problem (3.4) is already solved by some solution $\bar{y}$ obtained for a certain relaxation of Problem (3.4), where $\bar{y}_j$ might not be binary-valued for all $j \in J_2$. This can be achieved as follows. Let

$$\hat{Z}_s(\Omega) \equiv \{(x,y) : T_s x + W_s y \geq r_s, \text{ and } \alpha_s^k x + \beta_s^k y \geq \gamma_s^k, \forall k = 1, \ldots, K, x \in \Omega, y \geq 0\}, \quad (3.23)$$

where the constraints $y_j \leq 1$, $\forall j$, are included in $T_s x + W_s y \geq r_s$ and where $\alpha_s^k x + \beta_s^k y \geq \gamma_s^k, \forall k = 1, \ldots, K$, are a set of valid inequalities for $Z_s(\Omega)$. Given any $\bar{x} \in X \cap \Omega$, define the restricted set

$$\hat{Z}_{LB_s}(\bar{x}) \equiv \{y : (\bar{x}, y) \in \hat{Z}_s(\Omega)\}, \quad (3.24a)$$

along with the associated lower bounding value

$$\widehat{LB}_s(\bar{x}) \equiv \min\{g^T y : y \in \hat{Z}_{LB_s}(\bar{x})\}. \quad (3.24b)$$

Proposition 3.3.1. Given any $\bar{x} \in X \cap \Omega$, let $\bar{y}$ be an optimal solution that evaluates $\widehat{LB}_s(\bar{x})$. If $\bar{y}_j \in \{0,1\}$, $\forall j \in J_2$, or if $(\bar{x}, \bar{y})$ can be represented as a convex combination of some extreme points of $\hat{Z}_s(\Omega)$ such that these extreme points have binary y_j-variables for all $j \in J_2$, then $\bar{y}$ solves Problem (3.4) with $x \equiv \bar{x}$ fixed, i.e., $\widehat{LB}_s(\bar{x}) = LB_s(\bar{x})$.

Proof: Since $\hat{Z}_s(\Omega) \supseteq Z_s(\Omega)$, we have that

$$Z_{LB_s}(\bar{x}) \subseteq \hat{Z}_{LB_s}(\bar{x}), \text{ and hence, } LB_s(\bar{x}) \geq \widehat{LB}_s(\bar{x}). \quad (3.25)$$

Now, if $\bar{y}_j \in \{0,1\}$, $\forall j \in J_2$, then $(\bar{x}, \bar{y}) \in Z_s(\Omega)$. Alternatively, if $(\bar{x}, \bar{y})$ can be represented as a convex combination of some extreme points (x^p, y^p), $p \in P$, of $\hat{Z}_s(\Omega)$, where $y_j^p \in \{0,1\}$, $\forall j \in J_2$, $\forall p \in P$, we have that $(x^p, y^p) \in Z_s(\Omega)$, $\forall p \in P$, and so, $(\bar{x}, \bar{y}) \in Z_s(\Omega)$. In either case, this yields that $\bar{y} \in Z_{LB_s}(\bar{x})$, or that,

$$\widehat{LB}_s(\bar{x}) = g^T \bar{y} \geq \min_{y \in Z_{LB_s}(\bar{x})} g^T y = LB_s(\bar{x}). \quad (3.26)$$

The proof now follows from (3.25) and (3.26). □

We now present an algorithm that embeds the cutting plane game of Jeroslow (1980) along with a purification strategy within its operation to solve Problem (3.4).

Algorithm SP (*for solving Problem* (3.4)*, given* $\bar{x} \in X \cap \Omega$).

Initialization: Let $\alpha_s^0 x + \beta_s^0 y \geq \gamma_s^0$ be an initial set of valid inequalities for $Z_s(\Omega)$, which might be possibly empty or might be inherited from parent nodes (see Remark 3.3.1 below). Set

$$k = 0,\ T_s^0 = \begin{bmatrix} T_s \\ \alpha_s^0 \end{bmatrix},\ W_s^0 = \begin{bmatrix} W_s \\ \beta_s^0 \end{bmatrix}, \text{ and } r_s^0 = \begin{bmatrix} r_s \\ \gamma_s^0 \end{bmatrix}. \quad (3.27)$$

Step 1: Solve the LP relaxation

$$\widehat{LB}_s^k(\bar{x}) \equiv \min\{g^T y : W_s^k y \geq r_s^k - T_s^k \bar{x},\ y \geq 0\}, \tag{3.28}$$

and let $\bar{y}$ be an extreme point optimal solution. If $\bar{y}_j \in \{0,1\}$, $\forall j \in J_2$, then by Proposition 3.3.1, $\bar{y}$ solves Problem (3.4) and we can stop. Otherwise, denote $\hat{Z}_s(\Omega) \equiv \{(x,y) : T_s^k x + W_s^k y \geq r_s^k,\ x \in \Omega,\ y \geq 0\}$, and proceed to Step 2.

Step 2: Use the polynomial-time (purification) algorithm described in Sherali (1987) to represent $(\bar{x}, \bar{y})$ in terms of the extreme points of $\hat{Z}_s(\Omega)$. Let P be the index set of these extreme points, so that $(\bar{x}, \bar{y}) = \sum_{p \in P} \lambda_p (x^p, y^p)$, where $(x^p, y^p) \in vert(\hat{Z}_s(\Omega))$, $\forall p \in P$, and where $\sum_{p \in P} \lambda_p = 1$, $\lambda_p > 0$, $\forall p \in P$. If $y_j^p \in \{0,1\}$, $\forall j \in J_2$, $p \in P$, then again by Proposition 3.3.1, $\bar{y}$ solves Problem (3.4), and we can stop. Otherwise, replace $P \leftarrow P \setminus \{p \in P : y_j^p \in \{0,1\},\ \forall j \in J_2\} \neq \emptyset$, and go to Step 3.

Step 3: Denote

$$q = \max\{j \in J_2 : \exists p \in P, \text{ such that } 0 < y_j^p < 1\}. \tag{3.29}$$

Consider the subsystem extracted from $\hat{Z}_s(\Omega)$:

$$T_s^0 x + W_s^0 y \geq r_s^0 \tag{3.30a}$$
$$\alpha_s^l x + \beta_s^l y \geq \gamma_s^l, \quad \forall l \in \tau_{q-1} \tag{3.30b}$$
$$x \in \Omega,\ y \geq 0, \tag{3.30c}$$

where $\tau_{q-1} = \{l : \alpha_s^l x + \beta_s^l y \geq \gamma_s^l \text{ is a } j\text{-cut for } j \leq q-1\}$, and where a *j-cut* is a cut that has been generated during a previous iteration (of the form (3.33) as shown below), when j was identified as the selected index according to (3.29). Multiply all the constraints defining the system (3.30) by y_q and $1 - y_q$ and linearize applying RLT to obtain the following higher dimensional representation of $conv\{(x,y) :\ (3.30),\ y_q \text{ binary}\}$:

$$H_{2s}^k y + H_{3s}^k z \geq h_s^k - H_{1s}^k x,\ (y,z) \geq 0, \tag{3.31}$$

where z denotes the variable vector that results from the linearization of the products $y_j y_q$, $\forall j \neq q$, and $x_i y_q$, $\forall i$, and where we have set $y_q^2 = y_q$. Determine dual multipliers $\phi_s^k \geq 0$ for the structural constraints in (3.31) by solving the following separation problem to delete (x^t, y^t), where $t \in \underset{p \in P}{\operatorname{argmin}}\{|0.5 - y_q^p| : 0 < y_q^p < 1\}$.

$$\text{Minimize} \quad \phi_s^k(H_{2s}^k y^t) - \phi_s^k(h_s^k - H_{1s}^k x^t) \tag{3.32a}$$
$$\text{subject to} \quad \phi_s^k H_{3s}^k \leq 0 \tag{3.32b}$$
$$\phi_s^k(H_{2s}^k y^t) - \phi_s^k(h_s^k - H_{1s}^k x^t) \geq -1 \tag{3.32c}$$
$$\phi_s^k \geq 0, \tag{3.32d}$$

where (3.32c) serves to normalize the dual vector ϕ_s^k. Let $\bar{\phi}_s^k$ be a solution to (3.32). Generate a cut

$$\bar{\phi}_s^k(H_{2s}^k y) \geq \bar{\phi}_s^k(h_s^k - H_{1s}^k x), \tag{3.33}$$

which necessarily deletes the fractional extreme point (x^t, y^t) (see Sherali and Adams (1994) and Balas et al. (1993)). Label (3.33) as a q-cut, and denote it in the form of (3.30b), with $\alpha_s^{k+1} = \bar{\phi}_s^k H_{1s}^k$, $\beta_s^{k+1} = \bar{\phi}_s^k H_{2s}^k$, and $\gamma_s^{k+1} = \bar{\phi}_s^k h_s^k$. Let

$$T_s^{k+1} = \begin{bmatrix} T_s^k \\ \alpha_s^{k+1} \end{bmatrix},\ W_s^{k+1} = \begin{bmatrix} W_s^k \\ \beta_s^{k+1} \end{bmatrix},\ \text{and } r_s^{k+1} = \begin{bmatrix} r_s^k \\ \gamma_s^{k+1} \end{bmatrix}, \tag{3.34}$$

and increment k by one.

If $(\bar{x}, \bar{y})$ violates (3.33), return to Step 1. Otherwise, let $P \leftarrow P \setminus \{p \in P : (x^p, y^p)$ violates (3.33)$\}$. If $P \neq \emptyset$, then there exist other fractional vertices of $\hat{Z}_s(\Omega)$ that are not cut off by (3.33); hence, repeat Step 3. Otherwise, $P = \emptyset$, and the fractional extreme points of $\hat{Z}_s(\Omega)$ that were used to represent $(\bar{x}, \bar{y})$ in Step 2 are all cut off. Update $\hat{Z}_s(\Omega) = \{(x, y) : T_s^k x + W_s^k y \geq r_s^k,\ x \in \Omega,\ y \geq 0\}$. Since $(\bar{x}, \bar{y}) \in \hat{Z}_s(\Omega)$, it is still reproducible as a convex combination of the vertices of the updated $\hat{Z}_s(\Omega)$, and so, return to Step 2. □

Let K be the index for the last iteration, and denote by $\phi_s \geq 0$ an optimal dual solution to (3.28) where $k \equiv K$. We therefore obtain

$$\eta_s \geq \phi_s(r_s^K - T_s^K x) \tag{3.35}$$

as a Benders' cut of the type (3.11b) for scenario s, $s \in S$.

Proposition 3.3.2. Algorithm SP finitely solves Problem (3.4) for any fixed $x = \bar{x} \in X \cap \Omega$.

Proof: Algorithm SP terminates when the conditions stated in Proposition 3.3.1 are satisfied, whence, Problem (3.4) is solved; hence, we only need to show that it terminates finitely. System (3.30) is the intersection of the LP relaxation to the set $\Gamma_s(\Omega)$ that defines $Z_s(\Omega)$ in (3.3a, 3.3b) and the generated j-cuts for $j \leq q-1$, and system (3.32b)–(3.32d) is the reverse polar of the disjunctive set associated with (3.30) with y_q restricted to be 0 or 1 (see Sherali and Adams (1994)). The cuts (3.33) are constructed from the extreme points of this reverse polar. Following Theorem 3.1 of Balas et al. (1993), there are only a finite number of such cuts for each $j \in J_2$ that can be generated before we ultimately construct $Z_s(\Omega)$ in the worst case. At this point, the conditions of Proposition 3.3.1 must necessarily be satisfied, and so, the above procedure terminates finitely. This completes the proof. □

Remark 3.3.1. For a given Ω, cuts that are generated as above can be reused in the next call of the subproblem (3.4) while solving a given problem LBMP(Ω). Moreover, as shown below, the information generated while solving LBMP(Ω) for one Ω can be advantageously reused based on the structure of (3.3) and (3.4) for a subsequent Ω. Likewise, we can share cut information *between scenarios*, for any given Ω. To see this, notice first of all that $\Omega^{N+t} \subset \Omega^N$, $\forall t = 1, 2$, $\forall N$. Hence, inductively, for each $s \in S$, we have that the cuts (3.33) that are

generated for $Z_s(\Omega^N)$ are valid for all subsequent sets $Z_s(\Omega^M)$, where $M > N$ and M is a node of the subtree that is rooted at node N (so that, $\Omega^M \subset \Omega^N$). Consequently, these cuts can be included in (3.27) when applying Algorithm SP to solve Benders' subproblems (3.4) under scenario s in the process of solving LBMP(Ω^M). Similarly, for any $\Omega^M \subset \Omega^N$, any dual multiplier $\phi \in \Lambda_s^{\Omega_N}$ (appropriately augmented) corresponds to a feasible, though not necessarily extremal, solution to $\Lambda_s^{\Omega_M}$, where $\Lambda_s^{\Omega_N}$ and $\Lambda_s^{\Omega_M}$ denote the sets of dual multipliers feasible to Problem (3.4) when solving a subproblem under the restrictions $x \in \Omega_N$ and $x \in \Omega_M$, respectively. (Imagine as if the restrictions representing Ω^M are written as those present in Ω^N plus any additional constraints, and let the dual multipliers associated with the RLT convexification constraints generated off these additional constraints be zeros.) Benders' cuts (3.35) developed for Ω^N are therefore also valid for Ω^M. Section 3.5 provides our implementation details for this cut-inheritance idea. □

Furthermore, for any given Ω, it is advantageous if the dual multipliers generated for one scenario can be shared with other scenarios to generate valid cuts. We discuss this in the setting of fixed recourse (i.e., $W_s \equiv W$, $\forall s \in S$). To this end, suppose that we have generated a series of cuts for Problem (3.4) under scenario s when solving LBMP(Ω). Denote $\bar{\phi}_s(q)$ as the dual multiplier $\bar{\phi}_s^k$ that was used to generate a q-cut for Problem (3.4) under scenario s. Rewrite $\bar{\phi}_s(q)$ as $[\bar{\psi}_1(q), \bar{\psi}_2(q), \bar{\theta}(q)]$, where $\bar{\psi}_1(q)$ and $\bar{\psi}_2(q)$ are the dual multipliers associated with the original set of constraints $T_s x + Wy \geq r_s$ multiplied by $(1 - y_q)$ and y_q, respectively, and $\theta(q)$ is the dual multiplier associated with the RLT constraints produced by multiplying the remaining cut constraints in (3.30a)–(3.30c) with $(1 - y_q)$ and y_q, and then linearizing. We denote the q^{th} column of matrix W as W^q, and the matrix formed by the remaining columns of W as $\tilde{W}$. Correspondingly, denote variable y without the q^{th} element as $\tilde{y}$. We then have the following result.

Proposition 3.3.3. If

$$[\bar{\psi}_2(q) - \bar{\psi}_1(q)]T_{s'} \leq 0, \tag{3.36a}$$

$$\text{and} \quad [\bar{\psi}_2(q) - \bar{\psi}_1(q)]\tilde{W} \leq 0, \tag{3.36b}$$

then the following is a valid inequality for scenario s':

$$\bar{\psi}_1(q)\tilde{W}\tilde{y} + \bar{\psi}_1(q)r_{s'}y_q + \bar{\psi}_2(q)(W^q - r_{s'})y_q \geq \bar{\psi}_1(q)(r_{s'} - T_{s'}x). \tag{3.37}$$

Proof: Upon multiplying $T_{s'}x + Wy \geq r_{s'}$ by $(1 - y_q)$ and y_q, and linearizing applying RLT, we obtain:

$$\begin{bmatrix} T_{s'} \\ 0 \end{bmatrix} x + \begin{bmatrix} \tilde{W} \\ 0 \end{bmatrix} \tilde{y} + \begin{bmatrix} r_{s'} \\ W_q - r_{s'} \end{bmatrix} y_q + \begin{bmatrix} -T_{s'}, & -\tilde{W} \\ T_{s'}, & \tilde{W} \end{bmatrix} z \geq \begin{bmatrix} r_{s'} \\ 0 \end{bmatrix} \begin{matrix} \\ \end{matrix} \begin{matrix} \leftarrow \bar{\psi}_1^T(q) \\ \leftarrow \bar{\psi}_2^T(q) \end{matrix} \tag{3.38}$$

Surrogating (3.38) using the nonnegative multipliers $\bar{\psi}_1(q)$ and $\bar{\psi}_2(q)$ as shown, and using (3.36a) and (3.36b) along with $z \geq 0$, implies that (3.37) is valid for scenario s'. This completes the proof. □

Motivated by Proposition 3.3.3 and the C^3 theorem of Sen and Higle (2005) for fixed recourse stochastic programs, we are interested in simultaneously generating RLT cuts sharing common coefficient for y-variables for all scenarios. In the following we show that RLT cuts are the same as disjunctive cuts, and we can then directly apply the C^3 theorem to derive such valid inequalities.

Let $\tilde{\beta}$ be the coefficient associated with $\tilde{y}$, i.e., $\beta y \equiv \tilde{\beta}\tilde{y} + \beta_q y_q$. Let $x \in \Omega \equiv \{x | l \leq x \leq u\}$ at some node of the DBAB process. We assume all selected j-cuts are already included in $T_s x + Wy \geq r_s$ and omit the iteration index, the superscript k, for T, W, and r for notation simplicity. Upon multiplying the constraints

$$T_s x + \tilde{W}\tilde{y} + W^q y_q \geq r_s \text{ and } l \leq x \leq u \tag{3.39}$$

by y_q and $(1 - y_q)$, linearizing upon substituting the resulting nonlinear terms using $z^x \equiv x y_q$ and $z^y \equiv \tilde{y} y_q$, the resulting higher dimensional system (3.31) is as follows:

$$\begin{aligned}
& T_s z^x + \tilde{W} z^y \geq (r_s - W^q) y_q && \leftarrow \phi_0 && \text{(3.40a)}\\
& -T_s z^x - \tilde{W} z^y \geq r_s - T_s x - \tilde{W}\tilde{y} - r_s y_q && \leftarrow \phi_1 && \text{(3.40b)}\\
& z^x \geq l y_q && \leftarrow \phi_{xl0} && \text{(3.40c)}\\
& -z^x \geq -u y_q && \leftarrow \phi_{xu0} && \text{(3.40d)}\\
& -z^x \geq -l y_q + l - x && \leftarrow \phi_{xl1} && \text{(3.40e)}\\
& z^x \geq u y_q - u + x. && \leftarrow \phi_{xu1} && \text{(3.40f)}
\end{aligned}$$

The cut generation problem as a counterpart to (3.32) then takes the following form:

$$\begin{aligned}
\text{Minimize } & \tilde{\beta}_s \bar{\tilde{y}} + \beta_{qs} \bar{y}_q + \alpha_s \bar{x} - \gamma_s && \text{(3.41a)}\\
\text{subject to: } & (\phi_0 - \phi_1) T_s + \phi_{xl0} - \phi_{xu0} - \phi_{xl1} + \phi_{xu1} = 0 && \text{(3.41b)}\\
& (\phi_0 - \phi_1)\tilde{W} = 0 && \text{(3.41c)}\\
& \alpha_s = \phi_1 T_s + \phi_{xl1} - \phi_{xu1} && \text{(3.41d)}\\
& \tilde{\beta}_s = \phi_1 \tilde{W} && \text{(3.41e)}\\
& \beta_{qs} = \gamma_s + \phi_0 (W^q - r_s) - \phi_{xl0} l + \phi_{xu0} u && \text{(3.41f)}\\
& \gamma_s = \phi_1 r_s + \phi_{xl1} l - \phi_{xu1} u && \text{(3.41g)}\\
& \tilde{\beta}_s \bar{\tilde{y}} + \beta_{qs} \bar{y}_q + \alpha_s \bar{x} - \gamma_s \geq -1 && \text{(3.41h)}\\
& \phi \geq 0,\ \alpha_s, \tilde{\beta}_s, \beta_{qs}, \gamma_s \text{ unrestricted}, && \text{(3.41i)}
\end{aligned}$$

where $(\bar{x}, \bar{y})$ is the current solution having $\bar{y}_q$, $q \in J_b$, fractional, and where (3.41h) serves as a normalization constraint. Note that the resulting cut is of the form

$$\bar{\alpha}_s x + \bar{\tilde{\beta}}_s \tilde{y} + \bar{\beta}_{qs} y_q \geq \bar{\gamma}_s, \tag{3.42}$$

where $(\bar{\alpha}_s, \bar{\tilde{\beta}}_s, \bar{\beta}_{qs}, \bar{\gamma}_s, \bar{\phi})$ solves Problem (3.41).

Proposition 3.3.4. Problem (3.41) is equivalent to

$$\begin{aligned}
\text{Minimize} \quad & \tilde{\beta}_s\bar{\bar{y}} + \beta_{qs}\bar{y}_q + \alpha_s\bar{x} - \gamma_s && \text{(3.43a)}\\
\text{subject to:} \quad & \alpha_s \geq \phi_0 T_s + \phi_{xl0} - \phi_{xu0} && \text{(3.43b)}\\
& \alpha_s \geq \phi_1 T_s + \phi_{xl1} - \phi_{xu1} && \text{(3.43c)}\\
& \tilde{\beta}_s \geq \phi_0 \tilde{W} && \text{(3.43d)}\\
& \tilde{\beta}_s \geq \phi_1 \tilde{W} && \text{(3.43e)}\\
& \beta_{qs} \geq \gamma_s + \phi_0(W^q - r_s) - \phi_{xl0} l + \phi_{xu0} u && \text{(3.43f)}\\
& \gamma_s \leq \phi_1 r_s + \phi_{xl1} l - \phi_{xu1} u && \text{(3.43g)}\\
& \tilde{\beta}_s\bar{\bar{y}} + \beta_{qs}\bar{y}_q + \alpha_s\bar{x} - \gamma_s \geq -1 && \text{(3.43h)}\\
& \phi \geq 0,\ \alpha_s, \tilde{\beta}_s, \beta_{qs}, \gamma_s \text{ unrestricted.} && \text{(3.43i)}
\end{aligned}$$

Proof: Noting the objective coefficients of β_{qs} and γ_s in Problem (3.41), we can equivalently write the equalities of (3.41f) and (3.41g) as the inequalities (3.43f) and (3.43g), respectively, as these constraints will always hold as equalities in any optimal solution. Substituting for the terms involving ϕ_1 in (3.41d) and (3.41e) into (3.41b) and (3.41c), respectively, yields $\alpha_s = \phi_0 T_s + \phi_{xl0} - \phi_{xu0}$ and $\tilde{\beta}_s = \phi_0 \tilde{W}$, i.e., (3.43b) and (3.43d) in equality forms.

To complete the proof, we now show that at an optimal solution to (3.43), Constraints (3.43b)–(3.43e) will hold as equalities as in (3.41b)–(3.41e). Suppose that, on the contrary, we have an optimal solution $(\hat{\alpha}, \hat{\bar{\bar{\beta}}}, \hat{\beta}_{qs}, \hat{\gamma}, \hat{\phi}_{xl1}, \hat{\phi}_{xl0}, \hat{\phi}_-)$, where ϕ_- represent the vector ϕ except for the elements ϕ_{xl1} and $\hat{\phi}_{xl0}$, such that

$$\hat{\alpha}_{si} = \hat{\phi}_0 T_{si} + \hat{\phi}_{xl0i} - \hat{\phi}_{xu0i} > \hat{\phi}_1 T_{si} + \hat{\phi}_{xl1i} - \hat{\phi}_{xu1i}, \text{ or} \tag{3.44a}$$

$$\hat{\alpha}_{si} = \hat{\phi}_1 T_{si} + \hat{\phi}_{xl1i} - \hat{\phi}_{xu1i} > \hat{\phi}_0 T_{si} + \hat{\phi}_{xl0i} - \hat{\phi}_{xu0i} \tag{3.44b}$$

for some $i \in I$, where T_{si} is the i^{th} column of T_s.

If (3.44a) occurs, then let $\phi_{\epsilon i} = [\hat{\phi}_0 T_{si} + \hat{\phi}_{xl0i} - \hat{\phi}_{xu0i}] - [\hat{\phi}_1 T_{si} + \hat{\phi}_{xl1i} - \hat{\phi}_{xu1i}] > 0$. Obtain a new solution $(\hat{\alpha}, \hat{\bar{\bar{\beta}}}, \beta'_{qs}, \gamma'_s, \phi'_{xl1}, \hat{\phi}_{xl0}, \hat{\phi}_-)$ such that

$$\phi'_{xl1i} = \hat{\phi}_{xl1i} + \phi_{\epsilon i},\ \phi'_{xl1j} = \hat{\phi}_{xl1j},\ \forall j \neq i,\ \gamma'_s = \hat{\gamma}_s + l_i\phi_{\epsilon i}, \text{ and } \beta'_{qs} = \hat{\beta}_{qs} + l_i\phi_{\epsilon i}.$$

This new solution is feasible, satisfies $\hat{\alpha}_{si} = \hat{\phi}_0 T_{si} + \hat{\phi}_{xl0i} - \hat{\phi}_{xu0i} = \hat{\phi}_1 T_{si} + \phi'_{xl1i} - \hat{\phi}_{xu1i}$, and reduces the objective value by $\phi_{\epsilon i} l_i(1 - \bar{y}_q) > 0$. Hence, if (3.44a) holds, $(\hat{\alpha}, \hat{\bar{\bar{\beta}}}, \hat{\beta}_{qs}, \hat{\gamma}, \hat{\phi}_{xl1}, \hat{\phi}_{xl0}, \hat{\phi}_-)$ cannot be an optimal solution. On the other hand, suppose that (3.44b) occurs. Then let $\phi_{\epsilon i} = [\hat{\phi}_1 T_{si} + \hat{\phi}_{xl1i} - \hat{\phi}_{xu1i}] - [\hat{\phi}_0 T_{si} + \hat{\phi}_{xl0i} - \hat{\phi}_{xu0i}] > 0$. Similarly, we can obtain a new solution $(\hat{\alpha}, \hat{\bar{\bar{\beta}}}, \beta'_{qs}, \hat{\gamma}_s, \hat{\phi}_{xl1}, \phi'_{xl0}, \hat{\phi}_-)$ such that

$$\phi'_{xl0i} = \hat{\phi}_{xl0i} + \phi_{\epsilon i},\ \phi'_{xl0j} = \hat{\phi}_{xl0j},\ \forall j \neq i, \text{ and } \beta'_{qs} = \hat{\beta}_{qs} - l_i\phi_{\epsilon i}.$$

Again, this new solution is feasible, satisfies $\hat{\alpha}_{si} = \hat{\phi}_1 T_{si} + \hat{\phi}_{xl1i} - \hat{\phi}_{xu1i} = \hat{\phi}_0 T_{si} + \phi'_{xl0i} - \hat{\phi}_{xu0i}$, and reduces the objective value by $\phi_{\epsilon i} l_i \bar{y}_q > 0$. Hence, if (3.44b) holds, then $(\hat{\alpha}, \hat{\tilde{\beta}}, \hat{\beta}_{qs}, \hat{\gamma}, \hat{\phi}_{xl1}, \hat{\phi}_{-})$ cannot be an optimal solution, either. Therefore, we always have (3.43b) and (3.43c) tight at an optimal solution. Similarly, (3.43d) and (3.43e) always holds as equalities in an optimal solution. □

Now, consider the disjunctive cut generation where $Ax \geq b$ contains only the bounding constraints $l \leq x \leq u$ for x. Applying $y^s_q \in \{0, 1\}$ in the system (3.39), we obtain the following disjunction for scenario s:

$$\begin{array}{r} \phi_1 \rightarrow \\ \phi_{xl1} \rightarrow \\ \phi_{xu1} \rightarrow \\ \lambda_1 \rightarrow \end{array} \left\{ \begin{array}{l} T_s x + \tilde{W} \tilde{y}^s \geq r_s \\ x \geq l \\ -x \geq -u \\ -y^s_q \geq 0 \end{array} \right\} \bigvee \left\{ \begin{array}{l} T_s x + \tilde{W} \tilde{y}^s \geq r_s - W^q \\ x \geq l \\ -x \geq -u \\ y^s_q \geq 1 \end{array} \right\} . \begin{array}{l} \leftarrow \phi_0 \\ \leftarrow \phi_{xl0} \\ \leftarrow \phi_{xu0} \\ \leftarrow \lambda_0 \end{array} \tag{3.45}$$

Using the associated multipliers, a disjunctive cut can be generated by solving the following problem:

$$\begin{aligned} \text{Minimize} \quad & \tilde{\beta}_s \bar{\tilde{y}} + \beta_{qs} \bar{y}_q + \alpha_s \bar{x} - \gamma_s && \text{(3.46a)} \\ \text{subject to:} \quad & \text{constraints (3.43b)–(3.43e), (3.43h), and} && \text{(3.46b)} \\ & \beta_{qs} \geq -\lambda_1 && \text{(3.46c)} \\ & \beta_{qs} \geq \lambda_0 && \text{(3.46d)} \\ & \gamma_s \leq \phi_0 (r_s - W^q) + \phi_{xl0} l - \phi_{xu0} u + \lambda_0 && \text{(3.46e)} \\ & \gamma_s \leq \phi_1 r_s + \phi_{xl1} l - \phi_{xu1} u && \text{(3.46f)} \\ & \phi, \lambda \geq 0, \ \alpha_s, \tilde{\beta}_s, \beta_{qs}, \gamma_s \text{ unrestricted.} && \text{(3.46g)} \end{aligned}$$

Since β_{qs} appears only in constraints (3.46c) and (3.46d), we will have $\beta_{qs} \equiv \lambda_0 \geq -\lambda_1$ in an optimal solution. Constraints (3.46c)–(3.46e) then directly reduce to (3.43f), and Problem (3.46) is exactly the same as Problem (3.43). Therefore, the cut (3.42) generated using a RLT process is indeed also a disjunctive cut.

Applying the C^3 result from Sen and Higle (2005), we can then generate disjunctive cuts that are valid for the disjunctions (3.45) for all scenarios, so that the cuts contain a common coefficient β for y. Similar to (2.17), the cut coefficients can be obtained by collecting the

coefficient matrices of all scenarios and solving the following problem:

$$\text{Minimize} \quad \sum_s p_s(\tilde{\beta}\bar{\bar{y}}^s + \beta_q \bar{y}_q^s + \alpha_s \bar{x} - \gamma_s) \tag{3.47a}$$

$$\text{subject to:} \quad \alpha_s \geq \phi_h T_s + \phi_{xlhs} - \phi_{xuhs}, \quad \forall h = 0, 1, \ \forall s \in S \tag{3.47b}$$

$$\tilde{\beta} \geq \phi_h \tilde{W}, \quad \forall h = 0, 1 \tag{3.47c}$$

$$\beta_q \geq \gamma_s + \phi_0(W^q - r_s) - \phi_{xl0s} l + \phi_{xu0s} u, \quad \forall s \in S \tag{3.47d}$$

$$\gamma_s \leq \phi_1 r_s + \phi_{xl1s} l - \phi_{xu1s} u, \quad \forall s \in S \tag{3.47e}$$

$$\sum_s p_s(\tilde{\beta}\bar{\bar{y}}^s + \beta_q \bar{y}_q^s + \alpha_s \bar{x} - \gamma_s) \geq -1 \tag{3.47f}$$

$$\phi \geq 0, \ \alpha_s, \tilde{\beta}, \beta_q, \gamma_s \text{unrestricted}. \tag{3.47g}$$

For an optimal solution $(\bar{\bar{\beta}}, \bar{\beta}_q, \bar{\alpha}_s, \bar{\gamma}_s, \forall s)$, $\bar{\bar{\beta}}\tilde{y}^s + \bar{\beta}_q y_q^s \geq \bar{\gamma}_s - \bar{\alpha}_s x$, $\forall s \in S$ are then valid inequalities for disjunctions (3.45) for all scenarios $s \in S$, and they share the same coefficient $\bar{\beta} \equiv (\bar{\bar{\beta}}, \bar{\beta}_q)$ for y.

3.3.2 Benders' Scheme for Solving LBMP(Ω)

In this sub-section, we combine a decomposition/relaxation scheme with a B&B procedure to solve LBMP(Ω) for use in Algorithm DBAB (for any specified "Ω"). In this process, the master program is solved using B&B, and whenever a feasible solution that satisfies the integrality requirements of the first-stage variables is obtained, the corresponding subproblem is solved for each scenario to verify feasibility to the overall master program, and to generate Benders' cuts when needed. Note that in Algorithm DBAB, when we use $\overline{\text{LBMP}}(\cdot)$ instead of LBMP($\cdot$) to compute lower bounds, the same algorithm described below can be used, except that the integrality restriction on the first stage variables are relaxed, i.e., in essence, we assume that $I_2 \equiv \emptyset$.

Algorithm LBMP (*for solving LBMP*(Ω) *using a B&B and row-generation scheme*).

Step 0: Initialization Step. Initialize the incumbent solution (x^*, η^*) as null and set its objective value as $\nu^* = \infty$, respectively. Let the iteration counter $r = 1$, the number of nodes enumerated $N = 1$, the current node $\lambda(r) = 1$, the current node restrictions $PS_{\lambda(r)} = \emptyset$, and commence with the list of active nodes $L_r = \{1\}$. Let $k = 1, \ldots, K_s$, $\forall s \in S$, be the indices of the Benders' cuts inherited from the enclosing partition $\Omega' \supset \Omega$ ($K_s = 0$ if no such cut exists).

Step 1: Branching Step. If an optimal solution has been derived for the current node $\lambda(r)$ at some previous iteration, then denote it as $(\hat{x}^{\lambda(r)}, \hat{\eta}_s^{\lambda(r)}, \forall s)$. If no new Benders' cuts have

been added since that solution was obtained, then set the optimal solution $(x^{\lambda(r)}, \eta_s^{\lambda(r)}, \forall s) \equiv (\hat{x}^{\lambda(r)}, \hat{\eta}_s^{\lambda(r)}, \forall s)$ and let $\nu^{\lambda(r)}$ be its objective value; otherwise, solve

$$\begin{aligned}
\text{Minimize} \quad & c^T x + \sum_s p_s \eta_s && \text{(3.48a)}\\
\text{subject to} \quad & PS_{\lambda(r)} && \text{(3.48b)}\\
& \eta_s \geq \xi_s^k - \zeta_s^k x, \quad \forall k = 1, \ldots, K_s, \forall s \in S && \text{(3.48c)}\\
& \eta_s \geq -M, \quad \forall s \in S : K_s = 0 && \text{(3.48d)}\\
& x \in \bar{X} \cap \Omega, && \text{(3.48e)}
\end{aligned}$$

where M is chosen as a large number to bound (3.48) in case $K_s = 0$ for any s, and $\bar{X}$ is the LP relaxation of X defined in (3.1c). Let $(x^{\lambda(r)}, \eta_s^{\lambda(r)}, \forall s)$ be an optimal solution and let $\nu^{\lambda(r)}$ be its objective value. (If (3.48) is infeasible, then set $\nu^{\lambda(r)} = \infty$.)

If $x_i^{\lambda(r)} \in \{0, 1\}$, $\forall i \in I_2$, then proceed to Step 2 if $x^{\lambda(r)} \neq \hat{x}^{\lambda(r)}$, and go to Step 3 if $x^{\lambda(r)} = \hat{x}^{\lambda(r)}$. Otherwise, let $q \in \text{argmin}\{|0.5 - x_i^{\lambda(r)}| : i \in I_2\}$, $PS_{N+1} = PS_{\lambda(r)} \cap \{x_q = 0\}$ and $PS_{N+2} = PS_{\lambda(r)} \cap \{x_q = 1\}$. Replace $\lambda(r)$ within L_r by the new node indices $N + t$, $\forall t = 1, 2$. Re-solve (3.48) with (3.48b) replaced by PS_{N+t}, for $t = 1, 2$, respectively, to obtain ν^{N+t}, $\forall t = 1, 2$. Increment N by two, and go to Step 3.

Step 2: Cut Generation Step. Apply Algorithm SP to evaluate $LB_s(x^{\lambda(r)})$, $\forall s \in S$, using (any of) the previously generated cuts (3.33) with their right-hand-sides modified according to the current solution $x^{\lambda(r)}$. For each scenario s, $s \in S$, if $\eta_s^{\lambda(r)} < LB_s(x^{\lambda(r)})$, then derive the Benders' cut (3.35), denote this cut as $\eta_s \geq \xi_s^{K_s+1} - \zeta_s^{K_s+1} x$, and increment K_s by 1. Furthermore, if $c^T x^{\lambda(r)} + \sum_s p_s LB_s(x^{\lambda(r)}) < \nu^*$, then update the incumbent solution $(x^*, \eta_s^*, \forall s)$ and objective value ν^* with $(x^{\lambda(r)}, LB_s(x^{\lambda(r)}), \forall s)$ and $c^T x^{\lambda(r)} + \sum_s p_s LB_s(x^{\lambda(r)})$, respectively.

Step 3: Fathoming and Node Selection Step. Fathom any non-improving nodes by setting $L_{r+1} \leftarrow L_r \setminus \{\lambda \in L_r : \nu^\lambda + \epsilon \geq \nu^*\}$, where $\epsilon \geq 0$ is a chosen optimality tolerance. If $L_{r+1} = \emptyset$, then stop with the incumbent solution as (ϵ-)optimal. Otherwise, increment r by one, and select an active node $\lambda(r) \in \text{argmin}\{\nu^\lambda : \lambda \in L_r\}$. Return to Step 1. □

3.4 Illustrative Example

Consider the following example:

$$\textbf{SMIP:} \quad \text{Minimize} \quad -5x_1 - x_2 + \sum_{s=1}^{2} 0.5 f_s(x) \tag{3.49a}$$

$$\text{subject to} \quad -x_1 - x_2 \geq -1.5 \tag{3.49b}$$

$$0 \leq x_1 \leq 1, \quad x_2 \text{ binary}, \tag{3.49c}$$

$$\text{where} \quad f_s(x) = \text{minimum} \ -16y_1 - 19y_2 - 23y_3 - 28y_4 \tag{3.49d}$$

$$\text{subject to} \begin{bmatrix} -2y_1 - 3y_2 - 4y_3 - 5y_4 \\ -6y_1 - y_2 - 3y_3 - 2y_4 \end{bmatrix} \geq [r_s - T_s x] \tag{3.49e}$$

$$y_j \leq 1, \quad \forall j \in J \tag{3.49f}$$

$$y_1, y_2 \geq 0, \quad y_3, y_4 \text{ binary}, \tag{3.49g}$$

and where $[r_1 - T_1 x] = \begin{bmatrix} -5 + 0.3x_1 \\ -10 + 0.3x_2 \end{bmatrix}$ and $[r_2 - T_2 x] = \begin{bmatrix} -10 + 0.2x_1 \\ -5 + 0.2x_2 \end{bmatrix}$. Note that we have $\Omega = \{x \in R^2 : 0 \leq x \leq e\}$.

3.4.1 Applying Algorithm DBAB While Using LBMP(Ω) for Computing Bounds

At the initial node, consider the solution of LBMP(Ω) for the original set Ω using Algorithm LBMP. We first initialize (x^*, η^*) as null and $\nu^* = \infty$, and then proceed as follows:

Algorithm LBMP, Iteration $r = 1$**.** The Node-1 problem at Step 1 is given by:

$$\text{Minimize} \quad -5x_1 - x_2 + 0.5\eta_1 + 0.5\eta_2 \tag{3.50a}$$

$$\text{subject to} \quad -x_1 - x_2 \geq -1.5 \tag{3.50b}$$

$$\eta_s \geq -M, \quad \forall s = 1, 2 \tag{3.50c}$$

$$0 \leq x_i \leq 1, \quad \forall i = 1, 2, \tag{3.50d}$$

which yields the solution $(x_1^{\lambda(1)}, x_2^{\lambda(1)}, \eta_1^{\lambda(1)}, \eta_2^{\lambda(1)}) = (1, 0.5, -M, -M)$, with an objective value $\nu^{\lambda(1)} = -5.5 - M$. We branch on $x_2^{\lambda(1)}$ to create two new nodes, Node 2 and Node 3, having respective additional restrictions of $x_2 = 0$ and $x_2 = 1$. The optimal objective values obtained for these two node problems are $\nu^2 = -5 - M$ and $\nu^3 = -3.5 - M$. Node 2 is selected at Step 3 since $\nu^2 < \nu^3$, and we return to Step 1.

Algorithm LBMP, Iteration $r = 2$**.** We retrieve the current node solution $(x_1^{\lambda(2)}, x_2^{\lambda(2)}, \eta_1^{\lambda(2)}, \eta_2^{\lambda(2)}) = (1, 0, -M, -M)$ from Iteration 1, and proceed to Step 2.

1. **Algorithm SP, Scenario** $s = 1$**:** Evaluating $LB_1(x^{\lambda(2)})$ yields $y = [1, 0.9, 0, 0]^T$ and $LB_1(x^{\lambda(2)}) = -33.1 > \eta_1^{\lambda(2)}$, without the need for generating any cut in this process. A Benders' cut as in (3.35) is therefore derived using only constraints (3.49e) and (3.49f) for $s = 1$ as follows:

$$\eta_1 \geq -35 + 1.9x_1. \tag{3.51}$$

2. **Algorithm SP, Scenario** $s = 2$**:** Three RLT cuts, as given below, are sequentially generated to cut off fractional solutions:

$$-0.25y_2 - 0.25y_3 - 0.5y_4 \geq -0.75 \tag{3.52a}$$

$$-0.366y_2 - 0.488y_3 - 0.268y_4 \geq -0.878 + 0.024x_1, \text{ and} \tag{3.52b}$$

$$-0.167y_2 - 0.167_3 - 0.167y_4 \geq -0.333. \tag{3.52c}$$

The final solution that satisfies the binary constraints is $y = [0.333, 1, 0, 1]^T$, with the objective value $LB_2(x^{\lambda(2)}) = -52.333 > \eta_2^{\lambda(2)}$. Hence, a Benders' cut

$$\eta_2 \geq -52.333 + 0.533x_2 \tag{3.53}$$

is derived using cuts (3.52a)–(3.52c) and the original constraints (3.49e) and (3.49f) for $s = 2$.

The updated incumbent solution $(x^*, \eta^*) = (1, 0, -33.1, -52.333)$, and the incumbent objective value $\nu^* = -5 \times 1 + 0.5 \times (-33.1) + 0.5 \times (-52.333) = -47.717$. Node 2 is again selected at Step 3.

Algorithm LBMP, Iteration $r = 3$**.** The Node-2 problem is now re-solved after appending the derived Benders' cuts (3.51) and (3.53). The optimal solution is given by $(x_1^{\lambda(3)}, x_2^{\lambda(3)}, \eta_1^{\lambda(3)}, \eta_2^{\lambda(3)}) = (1, 0, -33.1, -52.333)$. Since the x-variable values remain the same and the objective value $\nu^{\lambda(3)} = -47.717 = \nu^*$, we fathom this node at Step 3, and then select the only remaining active node, Node 3, for the next iteration.

Algorithm LBMP, Iteration $r = 4$**.** The Node-3 problem, including the Benders' cuts (3.51) and (3.53), yields the solution $(x_1^{\lambda(4)}, x_2^{\lambda(4)}, \eta_1^{\lambda(4)}, \eta_2^{\lambda(4)}) = (0.5, 1, -34.05, -51.8)$, with an objective value $\nu^{\lambda(4)} = -46.425 > \nu^*$. Since $x_2^{\lambda(4)}$ is integral, we transfer to Step 2 to solve the subproblems $LB_s(x^{\lambda(4)})$, $\forall s = 1, 2$, while including the RLT cuts (3.52a)–(3.52c) in the subproblem for Scenario 2. This yields $y = [1, 0.95, 0, 0]^T$ for Scenario 1 and $y = [0.3, 1, 0, 1]^T$ for Scenario 2, both of which satisfy the binary requirements for the y-variables. (Note that had we not inherited the RLT cuts (3.52a)–(3.52c) at this step, Scenario 2 would have produced a fractional optimal solution $y = [0.083, 1, 0.433, 1]^T$. This turns out to require the generation of three other RLT cuts that are different from (3.52a)–(3.52c) in order to yield the solution $y = [0.3, 1, 0, 1]^T$.)

Further, $LB_1(x^{\lambda(4)}) = -34.05 = \eta_1^{\lambda(4)}$ and $LB_2(x^{\lambda(4)}) = -51.8 = \eta_2^{\lambda(4)}$, and hence, no more Benders' cuts need to be generated. Node 3 is non-improving and is thus fathomed. No more active nodes are left, and so, the incumbent solution x^* and its objective value ν^* obtained in Iteration 2 solve LBMP(Ω). Since $x^* \in vert(\Omega)$, we have by Proposition 3.2.2 that this also solves the original SMIP.

3.4.2 Applying Algorithm DBAB While Using $\overline{\text{LBMP}}(\Omega)$ for Computing Bounds

In lieu of solving LBMP(Ω) for computing bounds, suppose that we solve its LP relaxation $\overline{\text{LBMP}}(\Omega)$. The resulting computations proceed as follows.

DBAB - Iteration $k = 1$: We start implementing DBAB with the original hyperrectangle Ω.
Solving $\overline{\text{LBMP}}(\Omega)$ via Algorithm LBMP:

Algorithm LBMP, Iteration $r = 1$ for $\overline{\text{LBMP}}(\Omega)$:
The initial linear master program for $\overline{\text{LBMP}}(\Omega)$ yields the solution $(x_1^{\lambda(1)}, x_2^{\lambda(1)}, \eta_1^{\lambda(1)}, \eta_2^{\lambda(1)}) = (1, 0.5, -M, -M)$. Scenario 1 yields $y = (1, 0.9, 0, 0)$, and generates the Benders' cut (3.51). Scenario 2 yields a fractional solution $y = (0.122, 1, 0.389, 1)$, which is represented by two extreme points of $\hat{Z}_2(\Omega)$: $(0.144, 1, 0.378, 1)$ and $(0.1, 1, 0.4, 1)$. The second extreme point is used to generate an RLT cut. This process continues and sequentially generates five fractional solutions, each of which is representable by two extreme points of $\hat{Z}_2(\Omega)$, and is cut off by an RLT cut derived based on one of these extreme points. The final solution obtained is $y = (0.317, 1, 0, 1)$, and the Benders' cut generated using the original constraints (3.49e) and (3.49f) for $s = 1$ and the aforementioned five RLT cuts is given by (3.53).

Algorithm LBMP, Iteration $r = 2$ for $\overline{\text{LBMP}}(\Omega)$:
Appending the two Benders' cuts (3.51) and (3.53), and re-solving the master program, produces the solution $(x_1^{\lambda(2)}, x_2^{\lambda(2)}, \eta_1^{\lambda(2)}, \eta_2^{\lambda(2)}) = (1, 0.5, -33.1, -52.067)$, with an objective value -48.083. This turns out to be an optimal solution to $\overline{\text{LBMP}}(\Omega)$ because the objective values for the two subproblems obtained during the previous iteration are $LB_1(x^{\lambda(2)}) = -33.1 = \eta_1^{\lambda(2)}$ and $LB_2(x^{\lambda(2)}) = -52.067 = \eta_2^{\lambda(2)}$, respectively.

DBAB - Iteration $k = 2$: We continue Algorithm DBAB by partitioning Ω into $\Omega_1 = \{x \in R^2 : 0 \le x_1 \le 1, x_2 = 0\}$ and $\Omega_2 = \{x \in R^2 : 0 \le x_1 \le 1, x_2 = 1\}$, based on the fractional solution $x_2 = 0.5$ obtained for $\overline{\text{LBMP}}(\Omega)$.

Algorithm LBMP for Solving $\overline{\text{LBMP}}(\Omega_1)$ and $\overline{\text{LBMP}}(\Omega_2)$:
These master programs are re-solved after inheriting the Benders' cuts (3.51) and (3.53)

from Iteration 1 of Algorithm DBAB, and yield solutions $(1, 0, -33.1, -52.333)$ and $(0.5, 1, -34.05, -51.8)$, respectively, with the corresponding objective values -47.717 and -46.425. For each of $\overline{\text{LBMP}}(\Omega_1)$ and $\overline{\text{LBMP}}(\Omega_2)$, the subproblem solutions obtained for both the scenarios satisfy the binary restrictions with $LB_s(x^{\lambda(1)}) = \eta_s^{\lambda(1)}, \forall s$, and hence, we have obtained optimal solutions via the corresponding master problems. For $\overline{\text{LBMP}}(\Omega_1)$, since $x^{\lambda(1)} \in vert(\Omega_1)$, it also provides an incumbent solution x^* for the DBAB procedure, with an objective function value $\nu^* = -47.717$.

The node corresponding to Ω_2 in the DBAB tree is now fathomed since $\nu(\overline{\text{LBMP}}(\Omega_2)) = -46.425 > \nu^*$. Since no other active nodes exist, the incumbent solution x^* found when solving $\overline{\text{LBMP}}(\Omega_1)$ is an optimal solution.

3.5 Computational Results

The proposed decomposition-based B&B algorithm DBAB was implemented in C++ using the CPLEX Callable Library 8.1. The computations were carried out on a Sun-Blade-1000 (UltraSPARC-III) Workstation having 512 MB RAM and a cpu speed of 750 MHz. In our implementation, the maximum number of RLT cuts inherited by each node from its "ancestor" nodes was set to be 20–150 in order to limit the growth in the size of the problem. Whenever this limit was reached, the newly generated cuts were used to replace the cuts that were derived earliest, because the RLT cuts obtained higher in the tree are presumably weaker than the ones generated more recently. For benchmarking purposes, we also directly solved the deterministic equivalent problem (**DEP**) using CPLEX 8.1.

Our test problems comprise two groups of instances. The first group of instances have four first-stage variables and six to eight second-stage variables. They are similar to the example used in Section 3.4, with some additional variables and constraints. The first-stage problem has the following form:

$$\begin{aligned}
\text{Minimize} \quad & -2x_1 - 2.5x_2 - 2x_3 - 1.5x_4 + E[f_s(x)] && \text{(3.54a)}\\
\text{subject to} \quad & -x_1 - x_2 - x_3 - 0.5x_4 \geq -1.5 && \text{(3.54b)}\\
& -x_1 - 2x_3 - x_4 \geq -3 && \text{(3.54c)}\\
& -x_1 - 2x_2 - x_3 - 2x_4 \geq -5 && \text{(3.54d)}\\
& -2x_1 - x_3 - x_4 \geq -4 && \text{(3.54e)}\\
& 0 \leq x_1, x_2 \leq 1, \quad x_3, x_4 \text{ binary}, && \text{(3.54f)}
\end{aligned}$$

where constraints (3.54d) and (3.54e) were optionally excluded from some problems. The

second-stage problems were constructed as follows (having fixed recourse):

$$
\begin{aligned}
f_s(x) = \quad & \text{minimum} \ -16y_1 - 19y_2 - 23y_3 - 28y_4 - 15y_5 - 12y_6 - 10y_7 - 20y_8 && \text{(3.54g)}\\
& \text{subject to}\\
& -2y_1 - 3y_2 - 4y_3 - 5y_4 - y_5 - 2y_6 - y_7 - 2y_8 \geq -r_{s1} - \sum_{i\in I} T_{s1i}x_i && \text{(3.54h)}\\
& -6y_1 - y_2 - 3y_3 - 2y_4 - 2y_5 - y_6 - 2y_7 \geq -r_{s2} - \sum_{i\in I} T_{s2i}x_i && \text{(3.54i)}\\
& -3y_1 - 2y_2 - 5y_3 - y_4 - 3y_5 - y_6 - 3y_7 - 2y_8 \geq -r_{s3} - \sum_{i\in I} T_{s3i}x_i && \text{(3.54j)}\\
& -y_1 - 2y_2 - y_3 - 2y_4 - 3y_5 - 2y_6 - y_7 \geq -r_{s4} - \sum_{i\in I} T_{s4i}x_i && \text{(3.54k)}\\
& 0 \leq y_j \leq 1, \quad \forall j \in J_1, \quad y_j \text{ binary}, \forall j \in J_2. && \text{(3.54l)}
\end{aligned}
$$

Constraint (3.54k) and variables y_7 and y_8 were not included in all problems, and we defined $J_1 \equiv \{1, 2, 3\}$ for the problems having six second-stage variables, and $J_1 \equiv \{1, 2, 3, 4\}$ for the problems having eight second-stage variables. The technology matrices and right-hand-side values for the second-stage problems of this group were generated using uniform distributions over $[-0.3, 0]$ and $[-15, -5]$, respectively. For a given number of scenarios and a chosen number of variables and constraints, we generated twenty problem instances.

The second group of instances either have four first-stage variables and nine second-stage variables, with two binary variables in the first stage and four binary variables in the second stage, or have six first-stage variables and thirteen second-stage variables, with three binary variables in the first stage and six binary variables in the second stage. The coefficients and right-hand-side values are defined so that the solutions are non-trivial. For a given number of scenarios and a chosen number of variables and constraints, we generated eight problem instances.

Table 3.1 summarizes the number and the sizes of the test problems, along with the maximum, minimum, and average cpu times (in seconds) consumed when using CPLEX to directly solve the deterministic equivalent problem and when applying Algorithm DBAB. In DBAB_b, the binary restrictions on the first-stage variables are enforced when solving $\text{LBMP}(\cdot)$, and in DBAB_r, these restrictions are relaxed, i.e., $\overline{\text{LBMP}}(\cdot)$ is solved instead.

When solving DEP directly via CPLEX, for these 104 test problems using a relative tolerance level $\epsilon = 0.1\%$, 43 instances were not solved to ϵ-optimality within the specified two-hour computational time limit and 12 instances were not solved due to insufficient computer memory. Using DBAB, however, only five instances were not solved by DBAB_b and seven instances were not solved by DBAB_r, due to memory limitations. None of the solution times reached the two-hour time limit in either version of DBAB. As evident from Table 3.1, Algorithm DBAB exhibits a more robust and efficient performance than solving DEP directly via CPLEX, with DBAB_b being relatively faster as compared with DBAB_r. We add the

Table 3.1: Computational results obtained using DEP and Algorithm DBAB.

Stage 1	Stage 2		DEP cpu (s.)			$DBAB_b$ cpu (s.)			$DBAB_r$ cpu (s.)		
$\|I_1\|, \|I_2\|, \|C_1\|$	$\|J_1\|, \|J_2\|, \|C_2\|$	$\|S\|$	max	min	avg.†	max	min	avg.†	max	min	avg.†
2, 2, 2	3, 3, 3	128	7288*	2	1461	491	13	123	1203	55	377
2, 2, 4	4, 4, 4	128	699	0	64	189	22	86	1422	59	543
2, 2, 2	3, 3, 3	196	7550*	40	4059	1563	28	413	3724	64	1312
2, 2, 2	3, 3, 3	256	7490*	7240*	7389*	1991	63	749	3571	115	998
2, 2, 4	5, 4, 4	128	7638*	6716	7351*	1013	64	439	2304	101	925
3, 3, 4	7, 6, 4	128	–‡	–‡	–‡	1193‡	366‡	732‡	624‡	624‡	624‡
3, 3, 4	7, 6, 4	256	7534*	7505*	7518*	1950	58	747	3223	462	1733

(1) $|I_1|$, $|I_2|$, $|J_1|$, and $|J_2|$ denote the number of continuous and binary variables, respectively, in each of the two stages; $|C_1|$ and $|C_2|$ denote the respective number of constraints in the two stages, and $|S|$ denotes the number of scenarios.
(2) * A two-hour time limit was imposed on the computational time, which was checked at the end of each iteration loop.
(3) † Average of all the computational times at termination, including those obtained at the specified two-hour limit, but excluding those for the problems that terminated due to the memory limit.
(4) ‡ When the memory limit of the computer was reached, the best feasible solution, if available, was obtained. In this case, the computational times do not represent the optimal solution times. When solved using DEP, all instances in this group reached the memory limit. Only one instance was solved within the memory limit using $DBAB_r$; hence, the numbers shown represent the result for the same instance.

caveat here that while CPLEX is a commercial package, our implementation of DBAB was rather crude in terms of data structure and memory usage. Hence, the performance of DBAB can be further enhanced by a more sophisticated implementation.

3.6 Summary and Conclusions

This chapter focuses on solving two-stage stochastic programs having mixed-integer first- and second-stage variables. The proposed decomposition-based branch-and-bound algorithm (DBAB) adopts a hyperrectangular partitioning process in the projected space of the first-stage variables. Lower bounds for the nodal problems in the branch-and-bound tree are computed by applying a modified Benders' approach that coordinates a master program with lower-bounding scenario-based second-stage subproblems. Each of these subproblems is derived by sequentially constructing a certain partial convex hull representation of the two-stage solution space. We show that the convexification (RLT) cuts derived for a given hyperrectangle Ω at any node are reusable in subsequent solutions of the subproblems for each scenario at this node by updating the first-stage variable values. Furthermore, these cuts can be inherited by the subproblems of the children nodes of Ω. Likewise, the Benders' cuts derived for a given Ω are also inheritable by the lower bounding master programs solved for the children nodes of Ω in the enumeration tree for Algorithm DBAB. The overall process is

proven to converge to a global optimum for the underlying stochastic mixed-integer problem.

We have illustrated the proposed algorithm using a numerical example, and have reported encouraging preliminary computational results using some randomly generated instances. The results clearly exhibit the relative robustness and effectiveness of applying the proposed algorithm in contrast with using a commercial package (CPLEX 8.1) on a deterministic equivalent formulation of the stochastic program.

Chapter 4

Two-Stage Fleet Assignment Model Considering Stochastic Demands

4.1 Introduction

Due to high aircraft capital and operational costs, airlines need to utilize their equipment capacity efficiently by assigning aircraft judiciously to the different scheduled flight legs in order to accommodate path- or itinerary-based passenger demands to the extent possible so as to maximize profits. This decision process involves fleets of different *families* of aircraft each composed of multiple *types* of aircraft having specific capacities, and is known as the airline *fleet assignment problem* (FAP). Airlines typically need to solve this problem 10-12 weeks in advance because the crew scheduling decisions are governed by the type of aircraft used on the different legs, and these decisions need to be in place 8-10 weeks prior to departure by union regulations. Complicating this situation is the fact that airline companies face one of the most random markets among industries. At such an early stage, demand is highly uncertain. As a result, airlines conduct an initial fleet assignment based on limited demand projections, and as departures become more imminent and the demand forecast improves, they revise this assignment (4-6 weeks in advance via a so-called *re-fleeting* problem) to better match capacities of assigned flights with passenger demands. After this strategic re-fleeting stage, within about two weeks of departure, only limited aircraft swaps can be performed among restricted legs or routes that have the same origins and destinations. The procedures of initial fleet assignment, strategic re-fleeting, and swapping form a three-step dynamic fleet assignment framework.

In this Chapter, we focus on the first two steps of the aforementioned dynamic fleet assignment framework—the initial fleet assignment and the subsequent re-fleeting stage. We also integrate this fleeting decision with a component from yield management: to determine how many passengers to accept on the scheduled flight network. Most of the earlier FAM

formulations implicitly consider passenger demands to be independent between flight legs. However, a significant portion of demand served by US airlines consists of *multiple-leg passengers* (i.e., passengers whose itineraries are comprised of more than one leg), especially under the hub-and-spoke network structure, in which passengers typically fly into and out of hub airports en route from their origins to their destinations. Clearly, the demand coming from multiple-leg passengers will be dependent on the availability of seats on all the involved legs between their origins and destinations, and is known as the *network effect*. Similarly, passengers flying on a particular leg are *not identical* in terms of the revenue that they generate and the airline resources that they consume. Therefore, we use *path-based* (or *itinerary-based*) passenger demand information, rather than simply leg-based demand data, to accommodate this network effect.

Furthermore, as mentioned above, the crew scheduling decisions are affirmed based on the initial fleet assignment. Different fleet families usually have different crew requirements, and crew members are usually qualified to fly only the types of aircraft within a single family. Consequently, in order to retain the crew schedules undisturbed, flights can only be re-matched with aircraft types belonging to the same family as prescribed by the initial fleet assignment. For this reason, the initial fleet assignment constrains the later re-fleeting in the sense that the re-fleeting decisions have to be confined within the same fleet families. Furthermore, the capacities of the aircraft among the different fleet types within any given family are often not significantly different; the major capacity differences come from fleet types across different families. If the initial fleeting decision is based on expected demands related to a highly stochastic distribution, and the realized demands for some flights turn out to be significantly different from these expected values and lie beyond the capacity range of the assigned family, then the potential benefits of re-fleeting will be limited. For this purpose, we need to consider the re-fleeting opportunities up-front at the initial fleet assignment stage itself so that this initial decision can provide sufficient flexibility for the subsequent re-fleeting processes to accommodate demand fluctuations. To achieve this flexibility, the initial decision should embrace various market scenarios so that it incorporates a sufficient look-ahead feature.

We propose a two-stage stochastic programming approach in which the first stage is concerned with the initial fleet assignment problem, but unlike the traditional deterministic methodology, focuses on making only a family-level assignment to each flight leg in light of future uncertainty that is characterized by multiple potential demand scenarios. The second stage thus performs the detailed assignments of fleet types within the allotted family to each leg under each scenario while considering the corresponding path-based demands. In this fashion, the initial decision of what aircraft family should serve each flight leg accomplishes the purpose of facilitating the necessary crew scheduling decisions, while judiciously examining the outcome of future re-fleeting actions based on different possible demand scenarios. Hence, when the actual re-fleeting process is enacted several weeks later, this anticipatory initial family-level assignment will hopefully provide an improved overall fleet type re-allocation that better matches demand. We also propose several enhanced fleet

assignment models, including a robust optimization model that controls decision variations among scenarios and a stochastic programming model that considers the recapture effect. In addition to these modeling concepts and frameworks, our other contributions in this chapter include a polyhedral analysis for the basic model, the design of suitable solution approaches, and a computational study to demonstrate the efficacy of using the proposed methodology in contrast to the traditional deterministic approach, and the efficiency of the proposed algorithms as compared with directly solving the deterministic equivalent formulation of the developed stochastic model.

The remainder of this Chapter is organized as follows. In Section 4.2, we formulate our basic two-stage stochastic programming model along with several extensions and variations that consider simplified second stage decisions, robust optimization strategies, and demand recapture features. We also discuss an auxiliary model for extracting from the resulting solution for the underlying two-stage program an initial assignment of the aircraft fleet types. Benders' decomposition-based algorithms for solving two relaxations of the primary model are developed in Section 4.3. Section 4.4 provides computational results, compares the stochastic programming model with deterministic implementations, and illustrates the efficiency of the proposed algorithms. Section 4.5 concludes the chapter along with recommendations for future research directions.

4.2 Stochastic Fleet Assignment Models

The fleet assignment problem is typically formulated as a mixed-integer program (see Sherali et al. (2006) for an overview of such models), and demand data is incorporated in the model using expected values based on historical information and market forecasts. As discussed above, at this stage, because of the high level of uncertainty in the market, expected values may not adequately reflect the final demand realization. Hence, we explicitly consider several potential market scenarios in making the initial (family-level) decisions so as to provide the subsequent re-fleeting process with greater flexibility in achieving higher profits. This future scenario-based concept lends FAP naturally to a two-stage stochastic programming approach, as formulated in Section 4.2.1. We also briefly remark on modeling FAP using a robust optimization extension of stochastic programming, as well as on adopting a stochastic model for the re-fleeting phase. All these models assume that customers who are not accepted on their requested paths are lost. In reality, they may be *recaptured* on some other paths. In Section 4.2.2, we discuss the incorporation of the recapture effect in our model. Note that for crew scheduling purposes, only an assignment of aircraft families to flights is needed; however, some other subsequent decisions such as through flight considerations (Ahuja et al. (2002, 2003)) and yield management (Glover et al. (1982), Erdmann et al. (1997)) may still need fleet type information. Accordingly, because the aircraft type assignment solution obtained from the two-stage SMIP model is a "policy" that specifies only a separate assignment for each potential demand realization, we also propose in Section 4.2.3 a model to prescribe an

overall fleet type assignment that minimizes future re-fleeting costs.

4.2.1 Stochastic Fleet Assignment Models without Considering Recapture

We now formulate the initial fleet assignment problem that incorporates subsequent re-fleeting opportunities using a two-stage stochastic mixed-integer program (SMIP). This formulation is derived from the single scenario (expected demand), path-based fleet assignment and re-fleeting models in Barnhart et al. (2002) and Sherali et al. (2005), and is based on a time-space flight network structure depicted in Figure 4.1, which is typically used in formulating the FAM and represents the flight schedule decided by the proceeding airline scheduling process.

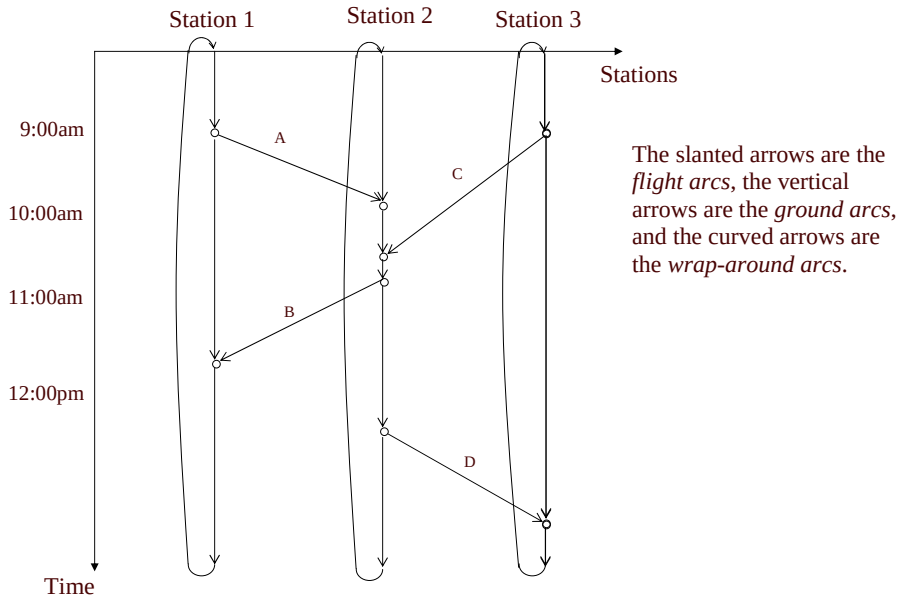

Figure 4.1: A three-station (airport) time-space flight network for a single aircraft type.

The overall time-space flight network representation superimposes a set of such networks, one for each fleet type in each family. This allows for fleet type-dependent flight-times and turn-times. In the subnetwork for any given fleet type, a vertical *network time-line* is associated with each station, consisting of a series of event nodes that sequentially occur at this station. These event nodes represent flight arrivals or departures at specific times. There are three types of arcs in the subnetwork for each fleet type: *ground arcs* representing aircraft staying at the same station for a given period of time, *flight arcs* representing flight legs, and *wrap-around arcs* (or *overnight arcs*) connecting the last event of the day with the first event of the day at each station, which accounts for a circulatory flow on a daily basis. Flight arcs connect the appropriate nodes on the different network time-lines for each fleet

type. For each station, the number of arriving flight arcs equals the number of leaving flight arcs, which is provided by the scheduling process to ensure the same daily schedule and to avoid deadheading. The requirement that only one fleet type (and thus only one fleet family) is assigned to each flight leg jointly regulates the flows that occur in the subnetworks of the different types, thus making all the subnetworks interdependent.

To formalize the model representation, consider the following notation for a two-stage SMIP formulation for the FAM (hereafter abbreviated as SPFAM):

Sets:

K **:** set of aircraft families in the fleet, indexed by k (e.g., $k = 1$ might denote the Boeing B737 family)

T_k **:** set of fleet types in family k, indexed by t (e.g., T_1 might include the types B737-300 and B737-500 in the Boeing B737 family)

T **:** set of all fleet types in the entire fleet, where $T = \cup_{k \in K} T_k$

N_t **:** set of nodes in fleet type t's network, $t \in T$; indexed by n

G_t **:** set of ground arcs in fleet type t's network, $t \in T$; indexed by g

L_k **:** set of flight legs assigned to family k in FAP (as a consequence of the to-be-determined initial fleeting decision), indexed by l

L **:** set of all flight legs, where $L = \cup_{k \in K} L_k$

Π **:** set of all identified paths in the flight schedule roster, indexed by i

$\Pi(l)$ **:** set of paths in Π that pass through leg l, $l \in L$

CS_t **:** set of arcs passing forward in time through a *counting time-line* (horizontal line corresponding to a fixed time) in aircraft type t's network, $t \in T$

S **:** set of scenarios corresponding to demand realizations, indexed by s.

Variables:

$$z_{lk} \text{ : } \begin{cases} 1, & \text{if flight leg } l \text{ is flown by aircraft family } k \text{ (in the first-stage problem), } l \in L,\ k \in K \\ 0, & \text{otherwise} \end{cases}$$

$$x_{lt}^{s} \text{ : } \begin{cases} 1, & \text{if flight leg } l \text{ is flown by fleet type } t \text{ (in the second-stage problem) under scenario } s, \\ & l \in L,\ t \in T,\ s \in S \\ 0, & \text{otherwise} \end{cases}$$

y_g^s **:** number of aircraft (of type t) on ground arc g in fleet type t's network for scenario s, $g \in G_t$, $t \in T$, $s \in S$

P_t^s **:** infeasibility penalty variable for the aircraft availability constraints for type t and scenario s, $t \in T$, $s \in S$. (This can also be interpreted as the extra number of aircraft chartered beyond the number of aircraft available, for type t, under scenario s.)

q_i^s **:** number of passengers flown (accepted demand) on path i under scenario s, $i \in \Pi$, $s \in S$.

Parameters:

μ_i^s **:** demand realization on path i, $i \in \Pi$, under scenario s, $s \in S$

μ^s **:** a vector of demand realization over all the paths in Π under scenario s, $s \in S$

$Q_s(z, \mu^s)$ **:** second-stage objective value for scenario s, given fleeting decision z from the first stage and demand realization μ^s, $s \in S$

c_{lt} **:** cost of covering leg l using fleet type t, $l \in L$, $t \in T$

Cap_t **:** capacity of fleet type t, $t \in T$

f_i^s **:** average fare price estimate on path i under scenario s, $i \in \Pi$, $s \in S$

p^s **:** probability for realization of scenario s, $s \in S$

A_t **:** number of available aircraft of type t, $t \in T$

ψ_t **:** an adequate cost for the penalty variable P_t^s used in the availability constraints of type t, $t \in T$. (This can also be interpreted as the chartering cost for using an extra aircraft for fleet type t, $t \in T$, or could be taken as an appropriately large penalty cost if no such opportunity exists.)

$bf_{ln} = \pm 1$, if flight l begins/ends at node n (in fleet type t's network), $l \in L$, $n \in N_t$, $t \in T$

$bg_{gn} = \pm 1$, if ground arc g begins/ends at node n (in fleet type t's network), $g \in G_t$, $n \in N_t$, $t \in T$.

The randomness is represented using a set of discrete scenarios S, in which each scenario $s \in S$ is associated with a realization of demand, μ^s, a corresponding fare, f^s, and a probability, p^s (where $\sum_{s \in S} p^s = 1$). The variable z is the aircraft family assignment decision, and x^s is the aircraft type assignment decision or, equivalently, the re-fleeting decision corresponding to each scenario $s \in S$. In practice, the demands are usually estimated using normal distributions. Because continuous distributions are usually much more difficult to handle computationally and algorithmically, the range of the possible demand values can be discretized to represent discrete scenarios, and the mean demand and probability for each

discretized segment are thus taken as the corresponding μ- and p-values, respectively. Depending on the level of discretization, this approach could generate far too many scenarios that need to be considered. Later, in Section 4.4.1, we design a scenario generation process that yields an adequate representation while keeping the problem size manageable. For now, we focus on modeling the problem.

In the first-stage of the proposed two-stage SPFAM, aircraft *families* are assigned to flight legs to minimize the expected second-stage fleeting costs. Given the assignment of aircraft families from the first-stage problem, the second-stage problem minimizes the net fleeting cost through an assignment of fleet types to flight legs and the consequent decision on the number of customers to accept for each path. The two-stage SPFAM can then be stated as follows.

Model 4.2.1. Two-Stage Stochastic Program FAM Considering Re-fleeting:

$$\textbf{SPFAM:} \qquad \text{Minimize} \sum_{s\in S} p^s Q_s(z,\mu^s) \tag{4.1a}$$

subject to

$$\sum_{k\in K} z_{lk} = 1, \quad \forall\ l \in L \tag{4.1b}$$

$$z \text{ binary}, \tag{4.1c}$$

where, for each $s \in S$, we have

$$Q_s(z,\mu^s) = \text{Minimum} \quad \sum_{l\in L}\sum_{t\in T} c_{lt}x^s_{lt} - \sum_{i\in\Pi} f^s_i q^s_i + \sum_{t\in T}\psi_t P^s_t \tag{4.1d}$$

subject to

$$\sum_{t\in T_k} x^s_{lt} = z_{lk}, \quad \forall\ l \in L, k \in K \tag{4.1e}$$

$$\sum_{l\in L} bf_{ln}x^s_{lt} + \sum_{g\in G_t} bg_{gn}y^s_g = 0, \quad \forall n \in N_t,\ \forall t \in T \tag{4.1f}$$

$$\sum_{l\in CS_t} x^s_{lt} + \sum_{g\in CS_t} y^s_g - P^s_t \le A_t, \quad \forall\ t \in T \tag{4.1g}$$

$$\sum_{i\in\Pi(l)} q^s_i \le \sum_{t\in T} Cap_t x^s_{lt}, \quad \forall\ l \in L \tag{4.1h}$$

$$q^s_i \le \mu^s_i, \quad \forall\ i \in \Pi \tag{4.1i}$$

$$x^s \text{ binary}, (y^s, q^s, P^s) \ge 0. \tag{4.1j}$$

The objective function (4.1a), in light of (4.1d), effectively maximizes the total expected profit. The first stage constraints (4.1b) assure that each flight leg is covered by exactly one aircraft family. Constraints (4.1e), (4.1f), and (4.1g) are the second-stage fleeting constraints, which respectively assure that each flight leg is covered by exactly one aircraft type from

the corresponding family that is assigned to it in the first stage, that the network flow is conserved, and that the number of aircraft used of each type does not exceed the available quantity (including any penalized excess used beyond A_t for type t). Constraints (4.1h) and (4.1i) are the second-stage *passenger mix constraints* that represent the capacity and demand restrictions, respectively.

Note that the solution to Model 4.2.1 yields an assignment of aircraft families to flight legs for the initial FAM and the alternative assignments of aircraft types under the different scenarios. These alternative type-level assignments can be viewed as potential decisions for the later strategic re-fleeting process. A family-level assignment to the legs does not decide on the capacities assigned to flight legs. Later, in Section 4.2.3, we present a technique for reconciling these scenario-based outcomes to prescribe an aggregate fleet-type decision in case this assigned capacity information is necessary for some subsequent processes, such as revenue management.

Alternatively, we can dispense with the scenario-based re-fleeting decisions from the second stage in favor of deriving more tractable, linear (as opposed to discrete) second-stage problems. In this case, the first-stage problem itself decides on the assignment of aircraft types to legs, and minimizes the cost of this assignment minus the expected revenue from the second stage. This leads to another stochastic variant of the initial FAM that considers path demands as follows.

Model 4.2.2. Two-Stage Stochastic Program FAM Precluding Re-fleeting:

$$\text{Minimize} \quad \sum_{l\in L}\sum_{t\in T} c_{lt}x_{lt} + \sum_{t\in T}\psi_t P_t - \sum_{s\in S} p^s Q_s^R(x,\mu^s) \tag{4.2a}$$

subject to

$$\sum_{t\in T} x_{lt} = 1, \quad \forall\ l \in L \tag{4.2b}$$

$$\sum_{l\in L} bf_{ln}x_{lt} + \sum_{g\in G_t} bg_{gn}y_g = 0, \quad \forall n \in N_t,\ \forall t \in T \tag{4.2c}$$

$$\sum_{l\in CS_t} x_{lt} + \sum_{g\in CS_t} y_g - P_t \le A_t, \quad \forall\ t \in T \tag{4.2d}$$

$$x \text{ binary}, (y,P) \ge 0, \tag{4.2e}$$

$$\text{where } Q_s^R(x,\mu^s) = \text{Maximum}\left\{ \sum_{i\in\Pi} f_i^s q_i^s \;\middle|\; \begin{array}{l} \sum_{i\in\Pi(l)} q_i^s \le \sum_{t\in T} Cap_t x_{lt}, \forall l \in L \\ 0 \le q_i^s \le \mu_i^s, \quad \forall i \in \Pi, \end{array} \right\}, \forall s \in S, \tag{4.2f}$$

and where $Q_s^R(x,\mu^s)$ is the second-stage objective (we use a superscript R to represent a restricted viewpoint that precludes re-fleeting). Compared with Model 4.2.1, Model 4.2.2 restricts one type-level assignment for all demand realizations; hence, the optimal objective value for Model 4.2.1 will be at least as good as that for Model 4.2.2. Furthermore, the

optimal type-level assignment obtained from Model 4.2.2 may be infeasible with respect to the family-level assignment obtained from Model 4.2.1. This is demonstrated in the following example, which also compares the stochastic and deterministic approaches.

Example 4.2.1. To illustrate a comparison of Models 4.2.1, 4.2.2, and a deterministic approach, we consider a hypothetical network having four legs that comprise two non-overlapping round-trips. Consider the following sets: $K = \{k_1, k_2\}$, $T_{k_1} = \{t_1, t_2\}$, $T_{k_2} = \{t_3, t_4\}$, $S = \{s_1, s_2\}$, and $\Pi = L = \{l_1, l_2, l_3, l_4\}$ (i.e., all paths contain only a single leg). We can only use the aircraft available (i.e., $\psi_t = \infty$, $\forall t \in T$, so that $P_t^s \equiv 0$, $\forall t \in T$, $s \in S$). Furthermore, let the costs, fares, and demand parameters be as specified in Table 4.1.

Table 4.1: Parameters for Example 4.2.1.

$l \in L$ or	Costs				Scenario 1		Scenario 2		mean	
$i \in \Pi$	c_{lt_1}	c_{lt_2}	c_{lt_3}	c_{lt_4}	f_i^1	μ_i^1	f_i^2	μ_i^2	$\bar{f}_i$	$\bar{\mu}_i$
l_1	9000	18000	19000	10000	400	90	970	140	628	110
l_2	9000	17000	18000	10000	430	100	1000	150	658	120
l_3	8000	17000	18000	9000	410	90	980	140	638	110
l_4	8000	18000	19000	9000	530	90	990	140	714	110

Also, suppose that each type has only one aircraft available, and that the capacities for the four types are 120, 138, 141, and 123, respectively. Let the probabilities for the two scenarios be $p^{s_1} = 0.6$ and $p^{s_2} = 0.4$, and thus the mean fare and demand, $\bar{f}$ and $\bar{\mu}$, take the values shown in the last two columns of Table 4.1. The assignments obtained from Models 4.2.1, 4.2.2, and a deterministic approach that uses the mean fare and demand values are displayed in Table 4.2.

Table 4.2: Fleet assignment solutions obtained via Models 4.2.1, 4.2.2, and a deterministic approach.

	Model 4.2.1			Model 4.2.2		Deterministic	
	family	type under s_1	type under s_2	family	type	family	type
l_1	k_2	t_4	t_3	k_2	t_4	k_1	t_1
l_2	k_2	t_4	t_3	k_2	t_4	k_1	t_1
l_3	k_1	t_1	t_2	k_2	t_4	k_1	t_1
l_4	k_1	t_1	t_2	k_2	t_4	k_1	t_1

The assignment obtained from Model 4.2.2 is to assign the single aircraft of type t_4 from family k_2 to all the legs, the assignment obtained from the deterministic approach is to assign the single aircraft of type t_1 from family k_1 to all the legs, while the assignment obtained from Model 4.2.1 utilizes aircraft from both families.

To compare the decision performance, let us take the fleet assignments from these three approaches, and examine the objective value achieved if the demand realization turns out to be scenarios 1 and 2, respectively. Table 4.3 summarizes the type assignment and the objective value obtained using the three approaches under these two scenarios.

Table 4.3: Performance of the three approaches under different scenarios.

	Model 4.2.1		Model 4.2.2		Deterministic	
	Scenario s_1	Scenario s_2	Scenario s_1	Scenario s_2	Scenario s_1	Scenario s_2
Objective	$-476,660$	$-446,620$	$-438,800$	$-127,600$	$-125,600$	$-129,600$
Expected	$-337,036$		$-318,212$		$-315,120$	

The expected objective values for Model 4.2.1, Model 4.2.2, and the deterministic approach are $-337,036$, $-318,212$, $-315,120$, respectively. Hence, while having a smaller problem size, both Model 4.2.2 and the deterministic approach lose certain optimality and flexibility features in regard to re-fleeting decisions. □

Remark 4.2.1. We can extend Model 4.2.1 to construct a Robust Optimization (RO) (see Mulvey et al. (1995)) formulation to stabilize the variability among the scenarios. The decision is more robust if the second-stage does not result in a wide range of decisions for different demand realizations. A mean/variance minimization approach is used for this purpose as formulated below, where the variability term is weighted by an appropriate scalar λ to compromise between the two objective terms.

Model 4.2.3. Robust Optimization Fleet Assignment Model:

$$\text{Minimize} \quad \sum_{s\in S} p^s \Big(\sum_{l\in L}\sum_{t\in T} c_{lt}x^s_{lt} + \sum_{t\in T}\psi_t P^s_t - \sum_{i\in\Pi} f^s_i q^s_i\Big)$$

$$+\lambda \sum_{s\in S} p^s \Big[\sum_{l\in L}\sum_{t\in T} c_{lt}\ \Big(x^s_{lt} - \sum_{s\in S} p^s x^s_{lt}\Big) - \Big(\sum_{i\in\Pi} f^s_i q^s_i - \sum_{s\in S} p^s \sum_{i\in\Pi} f^s_i q^s_i\Big)\Big]^2 \tag{4.3a}$$

$$\text{subject to constraints (4.1b), (4.1c), and (4.1e)-(4.1j),} \tag{4.3b}$$

where the second term in the objective function controls the variability due to the fleet type assignments and passenger mix decisions. □

Remark 4.2.2. The above models are for the initial FAP, which are to be solved well ahead of departure times. When departures become more imminent, a strategic re-fleeting process based on more accurate demand forecast may become necessary. At this time, the assignment of fleet types need to be confined within the same families, and also, we need to assign aircraft having sufficient capacities for the passengers who have already been accepted for the corresponding flights. A path-based stochastic programming re-fleeting

model similar to Model 4.2.2 can be solved at this stage, where attention is restricted to reassigning aircraft types within each family (separately), and where in (4.2f), the variables q_i^s are further restricted to be at least equal to the number of passengers already accepted on path i, $i \in \Pi$, $\forall s \in S$. Compared with the initial FAP, the re-fleeting model copes with less market uncertainty; hence, we can consider fewer scenarios at the second stage. □

4.2.2 SPFAM Second-Stage Problem Considering Recapture

So far, we have assumed that passengers either fly on the specific path that they have requested or they will be lost. In reality, however, a passenger, who cannot get on a desired first choice path, may accept another path between the same origin and destination or between some nearby cities. This kind of passenger rerouting is called *recapture* by the airlines. In the following, we discuss the modeling of this effect. Since the first-stage decisions do not involve the specific passenger acceptance decisions, we consider only the second-stage problem below, and drop the scenario superscript s for simplicity. We define β_{ij} as the proportion of *excess* (i.e., spilled) demand on path i that can be routed to a different path $j \in \Pi$, $j \neq i$. This is the proportion of passengers whose first choice is path i, but who can be routed to path j, in case their first choice is not available. Notice that path j usually has the same origin and destination as path i, but in general, it could originate and/or terminate at some stations that are respectively nearby to the origin and destination of path i. Hence, we let the parameters β_{ij} serve as *substitution factors* that control the possible rerouting decisions. These parameter values are inputs for the model. Accordingly, for each $i \in \Pi$, let us define $\Pi_i \equiv \{i\} \cup \{j \in \Pi, j \neq i : \beta_{ij} > 0\}$, and let the variable q_{ij}^r, for $j \in \Pi_i$, denote the demand on path i that is flown on path j when recapture is considered in the analysis. Thus, the second-stage problem is revised from Model 4.2.1 as shown below.

Model 4.2.4. Second-Stage Problem Considering Recapture (for each scenario):

$$\text{Minimize} \quad \sum_{l\in L}\sum_{t\in T} c_{lt}x_{lt} - \sum_{i\in\Pi}\sum_{j\in\Pi_i} f_j q^r_{ij} + \sum_{t\in T}\psi_t P_t + \delta \sum_{i\in\Pi}\sum_{\substack{j\in\Pi_i \\ j\neq i}} q^r_{ij}$$

subject to

$$\sum_{t\in T_k} x_{lt} = z_{lk}, \quad \forall\ l \in L, k \in K \tag{4.4a}$$

$$\sum_{l\in L} bf_{ln}x_{lt} + \sum_{g\in G_t} bg_{gn}y_g = 0, \quad \forall n \in N_t,\ \forall t \in T \tag{4.4b}$$

$$\sum_{l\in CS_t} x_{lt} + \sum_{g\in CS_t} y_g - P_t \leq A_t, \quad \forall\ t \in T \tag{4.4c}$$

$$\sum_{i\in\Pi}\sum_{j\in\Pi_i\cap\Pi(l)} q^r_{ij} \leq \sum_{t\in T} Cap_t x_{lt}, \quad \forall\ l \in\ L \tag{4.4d}$$

$$\sum_{j\in\Pi_i} q^r_{ij} \leq \mu_i, \quad \forall\ i \in\ \Pi \tag{4.4e}$$

$$q^r_{ij} \leq \beta_{ij}(\mu_i - q^r_{ii}), \quad \forall\ i \in\ \Pi, \quad j \in \Pi_i,\ j \neq i \tag{4.4f}$$

$$x \text{ binary},\ (y, q^r, P) \geq 0. \tag{4.4g}$$

In the objective function, we add a penalty term associated with a small positive coefficient δ for accepting path i passengers on path $j \in \Pi_i, j \neq i$, in order to prioritize the solutions that route passengers to their first choices. That is, among alternative optimal solutions to the problem without this term, the augmented objective function will prefer satisfying each path's demand on the path itself, and only when there is insufficient capacity on any path, will the excess path demand (spill) be routed to other paths having slack capacity. (Note that in lieu of using a single parameter δ, we could use path-specific parameters to compromise among alternative solutions.)

The penalty coefficient δ should be small enough so as not to impact the optimal solution to the original objective function. Sherali and Soyster (1983) have prescribed a set of weights that can be assigned to preemptively prioritize multiple objective functions so that the resulting nonpreemptive objective function produces an equivalent optimal solution to the underlying preemptive problem. In the context of Model 4.2.4, our first priority objective function is the cost expression $\sum_{l\in L}\sum_{t\in T} c_{lt}x_{lt} - \sum_{i\in\Pi}\sum_{j\in\Pi_i} f_j q^r_{ij}$. Let us denote this primary objective as $f_1(x, q)$. The second priority objective function is the penalty term $\sum_{i\in\Pi}\sum_{\substack{j\in\Pi_i \\ j\neq i}} q^r_{ij}$, which we denote as $f_2(q)$. To compute a value for δ that would enforce a preemptive priority among these two objective functions within Model 4.2.4, by Sherali and Soyster (1983), we need to first find the range of $f_2(q)$ over the feasible region, say, $f_{2min} \leq f_2(q) \leq f_{2max}$. Suppose further that f_1^* and $\hat{f}_1$ denote the minimum and the next lowest ($\neq f_1^*$) value taken

on by f_1 at the vertices to the convex hull of feasible solutions. Then, so long as δ satisfies

$$f_1^* + \delta f_{2max} < \hat{f}_1 + \delta f_{2min}, \tag{4.5}$$

we would have that Model 4.2.4 would produce an optimum where the value taken on by f_1 equals f_1^*. To see this, suppose on the contrary that an optimal solution to Model 4.2.4 yields $\tilde{f}_1$ and $\tilde{f}_2$ as the respective values of f_1 and f_2, realized at some vertex to the convex hull of feasible solutions, where $\tilde{f}_1 > f_1^*$. Consider any feasible solution that produces the value f_1^* for f_1, and let the corresponding value for f_2 in this solution be denoted as f_2^*. Then, by (4.5), we have

$$f_1^* + \delta f_2^* \leq f_1^* + \delta f_{2max} < \hat{f}_1 + \delta f_{2min} \leq \tilde{f}_1 + \delta \tilde{f}_2,$$

which contradicts the optimality of the solution yielding $\tilde{f}_1$ and $\tilde{f}_2$. Hence, by (4.5), it is sufficient to have

$$\delta < \frac{\hat{f}_1 - f_1^*}{f_{2max} - f_{2min}}. \tag{4.6}$$

Hence, we can prescribe δ to equal some lower bound for $\hat{f}_1 - f_1^*$ divided by some upper bound for $f_{2max} - f_{2min}$. Notice that the value of q_{ij}^r for $j \in \Pi_i$, $j \neq i$, ranges from 0 to $\min\{\beta_{ij}\,\mu_i, \max_{t \in T_k} Cap_t\}$. An upper bound of $f_{2max} - f_{2min}$ is therefore given by:

$$\sum_{i \in \Pi} \sum_{\substack{j \in \Pi_i \\ j \neq i}} \min\{\beta_{ij}\,\mu_i, \max_{t \in T_k} Cap_t\}.$$

To obtain a lower bound for the numerator in (4.6), notice that, as indicated by Sherali et al. (1988), the difference between unequal objective values corresponding to any pair of extreme points to the convex hull of feasible solutions (which are a subset of the extreme points to the LP relaxation) is bounded below by $|det_{max}|^{-2}$, where $|det_{max}|$ is the maximum absolute determinant of any basis to the LP relaxation after integerizing all the data. Theoretically, an upper bound for $|det_{max}|$ can be derived by considering the products of [1 + |each constraint coefficient appearing in the problem|]. This, however, may result in an extremely small value for δ. In practice, for Model 4.2.4, we can set a lower bound for $\hat{f}_1 - f_1^*$ to be 0.01, assuming that $f_1(x, q)$ takes on values having at most two significant decimal digits at extreme points to the convex hull of feasible solutions.

The formulation for Model 4.2.4 is similar to that proposed by Barnhart et al. (2002), with the exception that these authors focus on minimizing the number of spills. Define t_i^j, for $i \in \Pi$, $j \in \Pi, j \neq i$, as the number of passengers spilled from path i who are redirected to (but not necessarily accepted on) path j, and t_i^- as the number of passengers spilled from path i that are not recaptured on any other paths. Correspondingly, define the parameters b_i^j, for $i \in \Pi$, $j \in \Pi, j \neq i$, as the fraction of customers spilled from path i toward path j that are *successfully* captured on path j.

The Itinerary-based Fleet Assignment Model (IFAM) proposed by Barnhart et al. (2002) can be represented as follows:

$$\text{Minimize} \quad \sum_{l\in L}\sum_{t\in T} c_{lt}x_{lt} + \sum_{i\in\Pi}[f_i\, t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}}(f_i - b_i^j f_j)t_i^j]$$

subject to

$$\sum_{t\in T_k} x_{lt} = z_{lk}, \quad \forall\ l\in L, k\in K \tag{4.7a}$$

$$\sum_{l\in L} bf_{ln}x_{lt} + \sum_{g\in G_t} bg_{gn}y_g = 0, \quad \forall n\in N_t,\ \forall t\in T \tag{4.7b}$$

$$\sum_{l\in CS_t} x_{lt} + \sum_{g\in CS_t} y_g \leq A_t, \quad \forall\ t\in T \tag{4.7c}$$

$$\sum_{t\in T} Cap_t x_{lt} + \sum_{i\in\Pi(l)}(t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j) - \sum_{i\in\Pi}\sum_{\substack{j\in\Pi(l)\\ j\neq i}} b_i^j t_i^j \geq \sum_{i\in\Pi(l)}\mu_i, \quad \forall\ l\in\ L \tag{4.7d}$$

$$t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j \leq \mu_i, \quad \forall\ i\in\ \Pi \tag{4.7e}$$

$$x \text{ binary}, (y,t)\geq 0. \tag{4.7f}$$

The total lost revenue because of spill is $\sum_{i\in\Pi}(f_i t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}} f_i t_i^j)$, while the regained revenue because of recapture is $\sum_{i\in\Pi}\sum_{\substack{j\in\Pi\\ j\neq i}} b_i^j f_j t_i^j$. The difference between these two terms, namely, the second term in the objective function, is the net revenue loss due to spillage, which is to be minimized along with the operational cost $\sum_{l\in L}\sum_{t\in T} c_{lt}x_{lt}$. The constraints (4.7a), (4.7b), and (4.7c) are respectively identical to (4.4a), (4.4b), and (4.4c). In the capacity constraints (4.7d), for each leg l, the term $\sum_{i\in\Pi(l)}(t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j)$ is the number of passengers who requested but are spilled from the paths that contain leg l, and the term $\sum_{i\in\Pi}\sum_{\substack{j\in\Pi(l)\\ j\neq i}} b_i^j t_i^j$ is the number of passengers recaptured from other paths to the ones that contain leg l. The difference between these two terms is the net spillage from leg l. Therefore, constraints (4.7d) require that the capacity assigned to any leg cannot be less than the total demand on paths containing this leg minus the net spillage from this leg. Constraints (4.7e) are the demand constraints, which ensure that the spillage from path i does not exceed the demand on this path.

Now, suppose that we model our constraints (4.4d), (4.4e), and (4.4f) alternatively as

(4.7d) and (4.7e), re-written in a more evident form as follows:

$$\sum_{i\in\Pi(l)}[\mu_i - t_i^- - \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j] + \sum_{i\in\Pi}\sum_{\substack{j\in\Pi(l)\\ j\neq i}} b_i^j t_i^j \leq \sum_{t\in T} Cap_t x_{lt}, \quad \forall\, l \in\ L \tag{4.8a}$$

$$t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j \leq \mu_i, \quad \forall\, i \in\ \Pi \tag{4.8b}$$

$$(x,t) \geq 0. \tag{4.8c}$$

Also, for convenience, let us re-write (4.4d), (4.4e), and (4.4f) by defining q_{ij}^r for all $j \in \Pi$, as opposed to for all $j \in \Pi_i$, $\forall i$, with the implicit understanding that we would automatically have $q_{ij}^r = 0$, $\forall j \in \Pi \setminus \Pi_i$, $\forall i$, because $\beta_{ij} = 0$, $\forall j \in \Pi \setminus \Pi_i$, $\forall i$.

$$\sum_{i\in\Pi(l)} q_{ii}^r + \sum_{i\in\Pi}\sum_{\substack{j\in\Pi(l)\\ j\neq i}} q_{ij}^r \leq \sum_{t\in T} Cap_t x_{lt}, \quad \forall\, l \in\ L \tag{4.9a}$$

$$q_{ii}^r + \sum_{\substack{j\in\Pi\\ j\neq i}} q_{ij}^r \leq \mu_i, \quad \forall\, i \in\ \Pi \tag{4.9b}$$

$$q_{ij}^r \leq \beta_{ij}(\mu_i - q_{ii}^r), \quad \forall\, i \in\ \Pi,\ \forall j \in \Pi,\ j \neq i \tag{4.9c}$$

$$(x,q) \geq 0. \tag{4.9d}$$

Noting (4.8a) and (4.9a), it is evident that the following identities represent the basic relationship between the two models:

$$q_{ii}^r = \mu_i - t_i^- - \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j, \quad \forall i \in \Pi \tag{4.10a}$$

$$q_{ij}^r = b_i^j t_i^j, \quad \forall i \in \Pi, j \in \Pi, j \neq i. \tag{4.10b}$$

Hence, suppose that we have a feasible solution to (4.8a) – (4.8c). If we define the q-variables as per (4.10a) – (4.10b), then (4.9a), (4.9b), and (4.9d) are clearly satisfied. However, in order to satisfy (4.9c), we would need to have the following hold true:

$$b_i^j t_i^j \leq \beta_{ij}[t_i^- + \sum_{\substack{j\in\Pi\\ j\neq i}} t_i^j], \quad \forall i \in \Pi,\, \forall j \in \Pi,\, j \neq i. \tag{4.11}$$

Note that (4.11) requires a particular relationship for β_{ij} to satisfy in terms of the spillage accepted on path j from path i based on b_i^j, as a fraction of the total spillage from path i. Of course, this might not be always satisfied. For example, suppose that the solution from Model 4.2.4 has $\sum_{t\in T} Cap_t x_{lt} = 10$ for a given leg $l \in L$, and that the demand $\mu_1 = 40$ for

Path 1, which is assumed to be the only path containing leg l. In addition, suppose that Path 2 is the only path that can recapture passengers from Path 1, and that it has sufficient excess capacity. Then the solution from (4.8a) will yield $t_1^2 = 30$. If b_1^2 is specified as 0.5, then according to (4.10b), we will have $q_{12}^r = 15$. However, this will violate constraint (4.9c) if β_{12} is specified as a number smaller than 0.5.

Conversely, consider any feasible solution to (4.9a) – (4.9d). Depending on the b_i^j coefficients, we might not be able to find a corresponding t-solution to (4.10a) – (4.10b). However, for simplicity, if we assume that $b_i^j > 0 \ \forall i, j$, (it is sufficient to assume this for the $q_{ij}^r > 0$ variables) then (4.10a) – (4.10b) give

$$t_i^j = \frac{q_{ij}^r}{b_i^j}, \text{ and } t_i^- = \mu_i - \sum_{\substack{j \in \Pi \\ j \neq i}} (\frac{q_{ij}^r}{b_i^j}) - q_{ii}^r, \quad \forall i \in \Pi. \tag{4.12}$$

This solution satisfies (4.8a) and (4.8b), but in order to satisfy (4.8c), we would need t_i^- to be nonnegative in (4.12). Depending on the b_i^j-coefficients, this might not hold true. Take the previous example in which $\sum_{t \in T} Cap_t x_{lt} = 10$ for a given leg $l \in L$, and demand $\mu_1 = 40$ for Path 1. If we have $\beta_{12} = 0.8$, then the solution q_{11}^r could take the value 10, and q_{12}^r can take the value as large as 24 by constraint (4.9c). However, if b_1^2 were specified as a small fraction, 0.1, say, the IFAM model will not yield a feasible solution with $t_1^2 = 240$, because this would violate the nonnegativity constraint for t_1^-.

Noting from above that (4.9a) – (4.9d) offers an added flexibility in that any feasible solution to (4.8a) – (4.8c) would produce a feasible solution to (4.9a) – (4.9d) provided that the β-coefficients relate to the b-coefficients as per the reasonable relationship (4.11), whereas the converse requires more stringent conditions, we will opt to use (4.9a) – (4.9d) in our analysis. Besides, this affords a more direct designation and control of the recaptured spillage through the variables q_{ij}^r and the relationship (4.9b), and also conforms in structure with the other related models we consider, and thereby, provides a degree of uniformity in our overall approach.

We have so far presented several mathematical formulations (Models 4.2.1-4.2.4) for the FAM and re-fleeting process, which incorporate robust decision-making to reflect market randomness. Among these models, Model 4.2.1 is the focus of the remaining discussion.

4.2.3 Aircraft Type Assignment Strategies after Solving the SPFAM

In the proposed two-stage SPFAM, Model 4.2.1, the first-stage problem assigns aircraft families to flights, and constrained by the family assignment from the first stage, each second-stage subproblem assigns aircraft types to flight legs and determines accepted demands, depending on the demand scenario. The assignments of fleet types from the second stage

may vary from one scenario to another. Furthermore, if the problem is not solved to exact optimality, a binary fleet-type-assignment solution may not be available. However, a feasible and consistent fleet-type-assignment is often needed for the revenue management and other related decision systems. Therefore, in this section, we propose a strategy to reconcile the different assignments of fleet types resulting from the second-stage scenario-based decisions.

Let $(\bar{z}, \bar{x}, \bar{y}, \bar{q})$ denote the solution obtained for the SPFAM. In this solution, assume that $\bar{z}$ is binary, but $(\bar{x}, \bar{y})$ is not necessarily so, being perhaps a heuristic solution to the second-stage problem. Hence, given this solution, the flight legs covered by each aircraft family is decided, and the assignments of fleet types are to be determined based on the subnetworks comprised of flight legs that are covered by each family. The following proposed model solves a fleeting problem for each family $k \in K$, where the objective is to minimize the re-fleeting cost.

Define u_{lt} to be a binary decision variable that takes on a value of 1 if fleet type t is assigned to flight leg l, and 0 otherwise, $\forall l \in L,\ t \in T$. Also, let v_g be the number of aircraft on ground arc g, $\forall g \in G_t,\ t \in T$. Consider family $k \in K$. Let $c^l_{t,t'}$ be the cost associated with re-fleeting on leg l from an originally assigned type t' to a type t, $\forall l \in L_k,\ t,\ t' \in T_k$. This cost may include personnel, maintenance, and airport related costs (such as those incurred by gate changes), as well as costs incurred due to passenger dissatisfaction. The term $\sum_{s\in S} p^s \sum_{t'\in T_k} c^l_{t,t'} \bar{x}^s_{lt'}$ then represents the weighted average cost over the different scenarios to re-fleet leg l from an originally assigned type to a to-be-determined type t, $\forall l \in L_k,\ t \in T_k$. Accordingly, we solve the following model for each family $k \in K$, where constraints (4.13b)-(4.13e) are inherited from the single family FAM, and the objective function (4.13a) minimizes the total re-fleeting cost. The solution is an assignment of fleet types to legs that minimizes the future expected re-fleeting costs.

Model 4.2.5. Reconciling Fleet Type Assignments to Minimize Expected Re-fleeting Costs:

$$\text{Minimize} \quad \sum_{l\in L_k}\sum_{t\in T_k}\Big(\sum_{s\in S} p^s \sum_{t'\in T_k} c^l_{t,t'}\bar{x}^s_{lt'}\Big)u_{lt} + \sum_{t\in T_k}\psi_t P_t \tag{4.13a}$$

subject to

$$\sum_{t\in T_k} u_{lt} = 1, \quad \forall\ l \in L_k \tag{4.13b}$$

$$\sum_{l\in L_k} bf_{ln}u_{lt} + \sum_{g\in G_t} bg_{gn}v_g = 0, \quad \forall n \in N_t,\ \forall t \in T_k \tag{4.13c}$$

$$\sum_{l\in CS_t} u_{lt} + \sum_{g\in CS_t} v_g - P_t \le A_t, \quad \forall\ t \in T_k \tag{4.13d}$$

$$u \text{ binary}, (v, P) \ge 0. \tag{4.13e}$$

Remark 4.2.3. Consider the special case in which

$$c^l_{t,t'} = \begin{cases} \theta_l, & \text{if } t \ne t' \\ 0, & \text{if } t = t' \end{cases}, \quad \forall l \in L_k, t \in T_k, \tag{4.14}$$

where θ_l is some constant, $\forall l \in L_k$. In this case, the above objective function (4.13a) becomes,

$$\text{Minimize} \sum_{l\in L_k}\sum_{t\in T_k} u_{lt}[\sum_{s\in S} p^s \sum_{\substack{t'\in T_k \\ t'\neq t}} \theta_l \bar{x}^s_{lt'}] + \sum_{t\in T_k} \psi_t P_t. \tag{4.15a}$$

Using $\sum_{t'\in T_k} \bar{x}^s_{lt'} = 1$ from (4.1e), $\sum_{t\in T_k} u_{lt} = 1$ from (4.13b), and $\sum_{s\in S} p^s = 1$, (4.15a) reduces to

$$\text{Minimize} \sum_{l\in L_k} \theta_l - \sum_{l\in L_k}\sum_{t\in T_k} u_{lt}[\sum_{s\in S} \theta_l p^s \bar{x}^s_{lt}] + \sum_{t\in T_k} \psi_t P_t, \tag{4.15b}$$

which is equivalent to:

$$\text{Maximize} \sum_{l\in L_k}\sum_{t\in T_k} u_{lt}[\sum_{s\in S} \theta_l p^s \bar{x}^s_{lt}] - \sum_{t\in T_k} \psi_t P_t. \tag{4.15c}$$

An interesting special case arises when $\theta_l = \theta$, $\forall l \in L_k$, whence (4.15c) further reduces to

$$\text{Maximize} \sum_{l\in L_k}\sum_{t\in T_k} u_{lt}[\sum_{s\in S} p^s \bar{x}^s_{lt}] - \sum_{t\in T_k} \psi_t P_t. \tag{4.16}$$

In this case, the re-fleeting on all the legs from some originally assigned type to any other type is assumed to be equally undesirable. The term $\sum_{s\in S} p^s \bar{x}^s_{lt}$ thus denotes the probability that leg l will be assigned to type t, $\forall l \in L_k$, $t \in T_k$, and the term $\sum_{t\in T_k} u_{lt}[\sum_{s\in S} p^s \bar{x}^s_{lt}]$ then represents the probability that leg l retains its original assignment. Hence, the objective function (4.16) then maximizes the total probability of retaining the original assignment over all the legs. □

Remark 4.2.4. Consider another special case in which

$$c^l_{t,t'} = \begin{cases} b_l, & \text{if } t \neq t' \\ -a_l, & \text{if } t = t' \end{cases}, \quad \forall l \in L_k, t \in T_k, \tag{4.17}$$

where the coefficient a_l is a preference level or reward (negative cost) for retaining the original or initial fleet type assignment, and the coefficient b_l is a penalty for any change in this assignment, for each leg l. In this case, the objective function (4.13a) is equivalent to:

$$\text{Maximize} \sum_{l\in L_k}\sum_{t\in T_k} u_{lt}[\sum_{s\in S} p^s (a_l \bar{x}^s_{lt} - \sum_{\substack{t'\in T_k \\ t'\neq t}} b_l \bar{x}^s_{lt'}] - \sum_{t\in T_k} \psi_t P_t, \tag{4.18a}$$

or, using (4.1e) and (4.13b),

$$\text{Maximize} \sum_{l\in L_k}\sum_{t\in T_k} u_{lt}[\sum_{s\in S} p^s (a_l + b_l) \bar{x}^s_{lt}] - \sum_{t\in T_k} \psi_t P_t. \tag{4.18b}$$

We can then view $a_l + b_l$ as θ_l as given in (4.14), which again reduces the objective function to (4.15c), or to (4.16) for constant $a_l + b_l$ values, over $l \in L_k$. □

4.3 Solution Approach

Model 4.2.1 is a two-stage SMIP having binary variables in the first stage and binary and continuous variables in the second stage. Its focus is to assign aircraft families to the flight network with the purpose of retaining sufficient flexibility for later revisions on fleet type assignment. The number of daily flights and identified paths for a major airline can easily reach several thousands and tens of thousands, respectively. This dimension, coupled with a number of fleet types and duplications for multiple demand realizations for each path, poses a very large-scale problem. Careful solution approaches need to be designed to reduce the inherent combinatorial burden. In Section 4.3.1, we first discuss a Benders' decomposition based solution framework for a model that relaxes the binary restrictions in the second stage and exploits the partial separability with respect to aircraft families to generate cuts more efficiently. In Section 4.3.2, we then conduct a polyhedral analysis of the original model to promote binary-valued solutions in the second stage. Again, for this enhanced model, we relax the binary restrictions on the second stage x-variables and develop a Benders' decomposition-based solution procedure. The resulting algorithmic approaches are detailed in Section 4.3.3. We postpone the study of the extensions to robust optimization and recapture features as embodied in Models 4.2.3 and 4.2.4 to future research.

4.3.1 Decomposition-Based Framework

Sherali and Fraticelli (2002) have developed a modified Benders' decomposition algorithm for solving SMIPs in which the first-stage variables are binary-valued and the second stage involves mixed-integer 0-1 variables. Their solution process alternates between a master problem and subproblems corresponding to the different scenarios. The master problem is solved via an LP-based branch-and-bound/cut scheme in the space of the first-stage variables and some additional variables representing values of the second-stage functions. The subproblems are solved using a sequential convexification process (such as the Reformulation-Linearization Technique (RLT) process of Sherali and Adams (1990, 1994, 1999)), in terms of the recourse mixed-integer variables.

In order to ease the computational burden and retain a focus on prescribing family assignments via SPFAM, we solve a relaxed model SPFAM1, which is defined as SPFAM with the binary restrictions for the x-variables relaxed (but those on the z-variables retained), and with (4.1h) tightened to (4.19e) as follows:

$$\textbf{SPFAM1:} \qquad \text{Minimize} \sum_{s \in S} p^s Q_s(z, \mu^s) \tag{4.19a}$$

$$\text{subject to: (4.1b) and (4.1c),} \tag{4.19b}$$

where, for each $s \in S$,

$$Q_s(z, \mu^s) = \text{Minimum} \quad \sum_{l\in L}\sum_{t\in T} c_{lt}x^s_{lt} - \sum_{i\in\Pi} f^s_i q^s_i + \sum_{t\in T}\psi_t P^s_t \tag{4.19c}$$

$$\text{subject to} \quad \text{(4.1e)-(4.1g), and} \tag{4.19d}$$

$$\sum_{i\in\Pi(l)} q^s_i \le \sum_{t\in T} \tilde{c}^s_{lt}x^s_{lt}, \quad \forall\ l \in L \tag{4.19e}$$

$$q^s_i \le \mu^s_i, \quad \forall\ i \in \Pi \tag{4.19f}$$

$$(x^s, y^s, q^s, P^s) \ge 0. \tag{4.19g}$$

Here, in the restrictions (4.19e) that replace (4.1h), we have,

$$\tilde{c}^s_{lt} = \min\{Cap_t, \sum_{i\in\Pi(l)} \mu^s_i\}, \forall l \in L, t \in T, \tag{4.20}$$

where (4.19e) are valid and tighter inequalities based on the polyhedral analysis conducted by Sherali et al. (2005) on the deterministic version of a similar single-family model. Note that the uncertainty in SPFAM (Model 4.2.1) exists only in the second-stage objective function and the second-stage right-hand-sides. The use of (4.19e) and (4.20) in SPFAM1, however, introduces randomness into the recourse matrix, while tightening the model representation.

In the context of applying Benders' decomposition to SPFAM1, the master problem takes the form:

$$\text{Minimize} \quad \sum_{s\in S} p^s\eta^s \tag{4.21a}$$

$$\text{subject to}$$

$$\sum_{k\in K} z_{lk} = 1, \quad \forall\ l \in L \tag{4.21b}$$

$$\eta^s \ge \alpha^s - \gamma^s z, \quad \forall s \in S \tag{4.21c}$$

$$z \text{ binary}, \tag{4.21d}$$

which includes the first-stage constraints as well as some additional Benders' type of cuts (4.21c) generated from the subproblems as follows. In our implementation, when the branch-and-bound/cut process for the master problem determines a new incumbent binary-valued solution z, we solve the second-stage problem (i.e., the subproblem) under each scenario, which, by virtue of (4.1b) and using $\Pi = \bigcup_{k\in K}\bigcup_{l\in L_k} \Pi(l)$, partially decomposes into a collection of almost-separable single-family re-fleeting problems with some overlapping paths. Each of these models determines the corresponding aircraft type assignments to be made to the legs, L_k, allotted to the particular family k.

Specifically, given a binary solution $\bar{z}$ from (4.21), we can ascertain L_k, $\forall k \in K$. Then,

these partially decomposed problems take the following form, $\forall k \in K$, $s \in S$.

$$Q_s^k(z, \mu^s) = \text{Minimize} \sum_{l \in L_k} \sum_{t \in T_k} c_{lt} x_{lt}^s - \sum_{i \in \bigcup_{l \in L_k} \Pi(l)} f_i^s q_i^s + \sum_{t \in T_k} \psi_t P_t^s \tag{4.22a}$$

subject to

$$\sum_{t \in T_k} x_{lt}^s = \bar{z}_{lk} = 1, \quad \forall\, l \in L_k \qquad \leftarrow \phi_l^{1s} \tag{4.22b}$$

$$\sum_{l \in L_k} b f_{ln} x_{lt}^s + \sum_{g \in G_t} b g_{gn} y_g^s = 0, \quad \forall n \in N_t,\ \forall t \in T_k \qquad \leftarrow \phi_n^{2s} \tag{4.22c}$$

$$\sum_{l \in CS_t \cap L_k} x_{lt}^s + \sum_{g \in CS_t} y_g^s - P_t^s \le A_t, \quad \forall\, t \in T_k \qquad \leftarrow \phi_t^{3s} \tag{4.22d}$$

$$\sum_{i \in \Pi(l)} q_i^s \le \sum_{t \in T_k} \tilde{c}_{lt}^s x_{lt}^s, \quad \forall\, l \in L_k \qquad \leftarrow \phi_l^{4s} \tag{4.22e}$$

$$q_i^s \le \mu_i^s, \quad \forall\, i \in \cup_{l \in L_k} \Pi(l) \qquad \leftarrow \phi_i^{5ks} \tag{4.22f}$$

$$(x^s, y^s, q^s, P^s) \ge 0. \tag{4.22g}$$

To ease computational burden, instead of deriving Benders' cuts (4.21c) based on the subproblems (4.19c)-(4.19g) for each $s \in S$, we can solve the decomposed single-family problems (4.22) and derive Benders' cuts using optimal dual solutions obtained from these problems. However, since there exist overlapping paths among families, some of the constraints (4.19f) will be included in more than one of the constraints (4.22f) in these decomposed models, and may have different associated dual solutions. To reconcile these dual solutions, we can derive valid Benders' cuts for SPFAM1 as follows.

Proposition 4.3.1. Let the dual variables associated with (4.22b)-(4.22f) be as indicated alongside these equations, and let $\bar{\phi}$ be an optimal dual solution obtained for solving (4.22) for each $k \in K$. The following is a valid Benders' cut for SPFAM1:

$$\eta^s \ge \sum_{t \in T} A_t \bar{\phi}_t^{3s} + \sum_{i \in \Pi} \mu_i^s (\max_{k \in K^i} \bar{\phi}_i^{5ks}) + \sum_{k \in K} \sum_{l \in L_k} \bar{\phi}_l^{1s} z_{lk}, \tag{4.23}$$

where, given a master problem solution $\bar{z}$, we define $K^i \equiv \{k \in K \mid i \in \bigcup_{l \in L_k} \Pi(l)\}$, $\forall i \in \Pi$.

Proof: From dual feasibility for (4.22), we construct a dual feasible solution for the system (4.19d)-(4.19g). Let us first define $\bar{\phi}_l^{1s} \equiv 0$, $\forall l \in L \setminus \{L_k\}$, $\forall k \in K$. Then, given the dual solution to (4.22) for each $k \in K$, compose these together to consider the dual solution $\bar{\phi}^{1s}$, $\bar{\phi}^{2s}$, $\bar{\phi}^{3s}$, $\bar{\phi}^{4s}$, and $\bar{\phi}^{5s} \equiv (\max_{k \in K^i} \bar{\phi}_i^{5ks}, \forall i \in \Pi)$ associated respectively with the constraints (4.19d)-(4.19f) for the subproblem. Dual feasibility with respect to the columns of x^s, y^s, and P^s is evident from dual feasibility to (4.22). It remains to show that $\bar{\phi}_l^{4s}$, $\forall l \in L$ and $\max_{k \in K^i} \bar{\phi}_i^{5ks}$, $\forall i \in \Pi$, associated with (4.19e) and (4.19f), respectively, yield dual feasibility

with respect to the columns for q_i^s, $\forall i \in \Pi$. From the dual to (4.22), we have that for any $k \in K$,

$$\bar{\phi}_i^{5ks} + \sum_{\substack{l \in L_k: \\ i \in \Pi(l)}} \bar{\phi}_l^{4s} \leq -f_i^s, \quad \forall i \in \bigcup_{l \in L_k} \Pi(l). \tag{4.24a}$$

Since $\bar{\phi}_l^{4s} \leq 0$, $\forall l \in L$, we then have

$$\bar{\phi}_i^{5ks} + \sum_{\substack{l \in L: \\ i \in \Pi(l)}} \bar{\phi}_l^{4s} \leq -f_i^s, \quad \forall k \in K^i, i \in \Pi, \tag{4.24b}$$

and hence,

$$\max_{k \in K^i} \bar{\phi}_i^{5ks} + \sum_{\substack{l \in L: \\ i \in \Pi(l)}} \bar{\phi}_l^{4s} \leq -f_i^s, \quad \forall i \in \Pi. \tag{4.24c}$$

Therefore, the stated solution ($\bar{\phi}^{1s}$, $\bar{\phi}^{2s}$, $\bar{\phi}^{3s}$, $\bar{\phi}^{4s}$, $\bar{\phi}^{5s}$) is dual feasible to (4.19c)-(4.19g), and so (4.23) is a valid Benders' cut for SPFAM1. □

As far as a primal feasible solution is concerned, for updating incumbent values, denoting q_i^{sk} as the optimal value of q_i^s, $\forall i \in \cup_{l \in L_k} \Pi(l)$, obtained for the subproblem (4.22) solved for $k \in K$, we take $q_i^s = \min_{k \in K^i} q_i^{sk}$, $\forall i \in \Pi$, in the solution to (4.19c)-(4.19g).

Using Proposition 4.3.1, we will develop a Benders' decomposition based solution method (Algorithm 4.3.1) for this problem in Section 4.3.3. The fractional x-solutions will be reconciled as discussed in Section 4.2.3.

To obtain solutions even closer to the ones for SPFAM (Model 4.2.1), we next perform a polyhedral analysis in Section 4.3.2, which accommodates the binary restrictions on the x-variables in order to derive an enhanced model representation for SPFAM. In this resulting model, we will again relax x to be continuous-valued to obtain a model SPFAM2, and we will then apply Benders' partitioning to SPFAM2 in Section 4.3.3.

4.3.2 Polyhedral Analysis: Derivation of SPFAM2

Sherali et al. (2005) have performed a polyhedral analysis of the deterministic version of single-family models similar to the ones considered herein, and have proposed several reformulation and partial convex hull construction mechanisms, along with various classes of valid inequalities to tighten the model representation. In particular, they establish the following

result. For any $k \in K$ and $l \in L_k$ (based on $z_{lk} = 1$), let

$$
\begin{aligned}
Z^s_{lk} = \{ \quad & (x^s_{lt} \text{ for } t \in T_k,\ q^s_i \text{ for } i \in \Pi(l)) : \\
& \sum_{t \in T_k} x^s_{lt} = 1 && \text{(4.25a)} \\
& \sum_{i \in \Pi(l)} q^s_i \le \sum_{t \in T_k} \tilde{c}^s_{lt}\, x^s_{lt} && \text{(4.25b)} \\
& q^s_i \le \mu^s_i, \quad \forall i \in \Pi(l) && \text{(4.25c)} \\
& q^s \ge 0, \quad x^s \text{ binary}\},\ \forall s \in S. && \text{(4.25d)}
\end{aligned}
$$

The convex hull of Z^s_{lk} is then given as follows in a higher dimensional space that includes new variables Q^s_{ilt}, $\forall i \in \Pi(l)$, $t \in T_k$.

$$
\begin{aligned}
conv(Z^s_{lk}) \equiv \quad & \{(x^s_{lt} \text{ for } t \in T_k,\ q^s_i \text{ for } i \in \Pi(l)) : \\
& \textstyle\sum_{t \in T_k} x^s_{lt} = 1 && && \text{(4.26a)} \\
& \textstyle\sum_{i \in \Pi(l)} Q^s_{ilt} \le \tilde{c}^s_{lt}\, x^s_{lt}, && \forall\ t \in T_k && \text{(4.26b)} \\
& Q^s_{ilt} \le \mu^s_i\, x^s_{lt}, && \forall\ i \in \Pi(l),\ t \in T_k && \text{(4.26c)} \\
& q^s_i = \textstyle\sum_{t \in T_k} Q^s_{ilt}, && \forall\ i \in \Pi(l) && \text{(4.26d)} \\
& x^s \ge 0,\ Q^s \ge 0\}. && && \text{(4.26e)}
\end{aligned}
$$

We now examine a similar sub-structure, but one that includes the first- and second-stage variables for a given scenario. Specifically, consider the following subsystem extracted from SPFAM (Model 4.2.1) for each $l \in L$ and $s \in S$, where Cap_t is replaced with $\tilde{c}^s_{lt}$, $\forall t$, as in (4.19e) and (4.20):

$$
\begin{aligned}
Z^s_l = \quad & \{(z_{lk} \text{ for } k \in K, x^s_{lt} \text{ for } t \in T,\ q^s_i \text{ for } i \in \Pi(l)) : \\
& \textstyle\sum_{k \in K} z_{lk} = 1 && && \text{(4.27a)} \\
& \textstyle\sum_{t \in T_k} x^s_{lt} = z_{lk}, && \forall k \in K && \text{(4.27b)} \\
& \textstyle\sum_{i \in \Pi(l)} q^s_i \le \sum_{t \in T} \tilde{c}^s_{lt}\, x^s_{lt} && && \text{(4.27c)} \\
& q^s_i \le \mu^s_i, && \forall i \in \Pi(l) && \text{(4.27d)} \\
& q^s \ge 0, \quad (x^s, z) \text{ binary}\}. && && \text{(4.27e)}
\end{aligned}
$$

The convex hull of Z^s_l can then be constructed as follows.

Proposition 4.3.2. Let Z_l^s be given by (4.27) for any $l \in L$. Then, $conv(Z_l^s) = Z_l^{sc}$, where

$$
\begin{aligned}
Z_l^{sc} \equiv \ \{(z_{lk} \text{ for } k \in K, x_{lt}^s \text{ for } t \in T,\ q_i^s \text{ for } i \in \Pi(l)) : & \\
\textstyle\sum_{k \in K} z_{lk} = 1 & && \text{(4.28a)}\\
\textstyle\sum_{t \in T_k} x_{lt}^s = z_{lk}, & \quad \forall k \in K && \text{(4.28b)}\\
\textstyle\sum_{i \in \Pi(l)} Q_{ilt}^s \le \tilde{c}_{lt}^s \, x_{lt}^s, & \quad \forall\ t \in T && \text{(4.28c)}\\
Q_{ilt}^s \le \mu_i^s \, x_{lt}^s, & \quad \forall\ i \in \Pi(l),\ t \in T && \text{(4.28d)}\\
\textstyle\sum_{t \in T} Q_{ilt}^s = q_i^s, & \quad \forall\ i \in \Pi(l) && \text{(4.28e)}\\
x^s \ge 0,\ Q^s \ge 0\}. & && \text{(4.28f)}
\end{aligned}
$$

Proof: Note that (4.27a) and (4.27b) can be written equivalently as follows:

$$
\begin{aligned}
z_{lk} = \textstyle\sum_{t \in T_k} x_{lt}^s, \quad \forall k \in K && \text{(4.29a)}\\
\textstyle\sum_{t \in T} x_{lt}^s = 1. && \text{(4.29b)}
\end{aligned}
$$

Noting that the system (4.29b) and (4.27c–4.27e) has the same form as Z_{lk}^s in (4.25a–4.25d), and that (4.29a) simply equates each z_{lk} variable to an expression in the x-variables that is automatically binary valued for binary x, the result follows from the corresponding representation of $conv(Z_{lk}^s)$ given by (4.26a–4.26e). □

Remark 4.3.1. In cases where including Z_l^{sc} for even a few selected legs results in a prohibitive increase in problem size, we can derive valid RLT or projected cuts implied by this set as and when needed via a separation problem as follows. Denote $\xi^s \equiv (z, x^s, q^s)$, and let (4.28) be written more compactly as:

$$
\begin{aligned}
W_1 \xi^s + W_2 Q^s = h_1 \quad &\leftarrow \psi_1 && \text{(4.30a)}\\
W_3 \xi^s + W_4 Q^s \le 0 \quad &\leftarrow \psi_2 && \text{(4.30b)}\\
(\xi^s, Q^s) \ge 0. & && \text{(4.30c)}
\end{aligned}
$$

Given some fractional solution $\bar{\xi}^s$, in order to possibly delete this by a valid inequality in the space of the ξ^s-variables that is implied by (4.30), we can solve the following separation problem.

$$
\begin{aligned}
\text{Maximize} \quad & [\psi_1^T W_1 + \psi_2^T W_3]\bar{\xi}^s - \psi_1^T h_1 && \text{(4.31a)}\\
\text{subject to} & \\
& \psi_1^T W_2 + \psi_2^T W_4 \ge 0, && \text{(4.31b)}\\
& [\psi_1^T W_1 + \psi_2^T W_3]\bar{\xi}^s - \psi_1^T h_1 \le 1, && \text{(4.31c)}\\
& \psi_1 \text{ unrestricted}, \ \psi_2 \ge 0, && \text{(4.31d)}
\end{aligned}
$$

where (4.31c) is a normalization constraint. If the objective value of this problem is positive (i.e., equals 1 due to (4.31c)), then we will have obtained an RLT cut

$$
[\bar{\psi}_1^T W_1 + \bar{\psi}_2^T W_3]\xi^s \le \bar{\psi}_1^T h_1 \qquad \text{(4.32)}
$$

that cuts off $\bar{\xi}^s$, where $\bar{\psi}$ is an optimal solution to Problem (4.31). □

Next, we propose another strategy based on the result of Proposition 4.3.2, which is motivated by the fact that the set of additional variables Q^s_{ilt}, $\forall i \in \Pi$, $l \in L$, $t \in T$, $s \in S$, increases the problem size tremendously. In effect, we construct an alternative subsystem that introduces fewer additional variables into the formulation by suitably aggregating the constraints in Z^{sc}_l. Toward this end, for each $l \in L$ and $s \in S$, let us surrogate each of (4.28c) and (4.28d) over $t \in T_k$, $\forall k \in K$, and then substitute $Q^{zs}_{ilk} = \sum_{t \in T_k} Q^s_{ilt}$, $\forall i \in \Pi(l)$, $k \in K$, to obtain the following set.

$$
\begin{aligned}
\tilde{Z}^s_l = \ \{(z_{lk} \text{ for } k \in K, x^s_{lt} \text{ for } t \in T,\ q^s_i \text{ for } i \in \Pi(l)) :& \\
\textstyle\sum_{k \in K} z_{lk} = 1 & & \text{(4.33a)} \\
\textstyle\sum_{t \in T_k} x^s_{lt} = z_{lk}, & \forall k \in K & \text{(4.33b)} \\
\textstyle\sum_{i \in \Pi(l)} Q^{zs}_{ilk} \le \sum_{t \in T_k} \tilde{c}^s_{lt} x^s_{lt}, & \forall\ k \in K & \text{(4.33c)} \\
Q^{zs}_{ilk} \le \mu^s_i\, z_{lk}, & \forall\ i \in \Pi(l),\ k \in K & \text{(4.33d)} \\
\textstyle\sum_{k \in K} Q^{zs}_{ilk} = q^s_i, & \forall\ i \in \Pi(l) & \text{(4.33e)} \\
x^s \ge 0,\ Q^{zs} \ge 0\}. & & \text{(4.33f)}
\end{aligned}
$$

We now tighten $\tilde{Z}^s_l$ through an RLT lifting by multiplying each of (4.33b) and (4.33c) by its corresponding z_{lk}, and multiplying (4.33a) by x^s_{lt}, $\forall t \in T$, while retaining (4.33a) itself, and then substituting $V^s_{ltk} = x^s_{lt} z_{lk}$, $\forall t \in T_k$, $k \in K$, and recognizing that for binary z, we have, $x^s_{lt} z_{lk'} = 0$, $\forall t \in T_k$, $\forall k' \in K$, $k' \ne k$, and $Q^{zs}_{ilk} z_{lk} = Q^{zs}_{ilk}$, $\forall i \in \Pi(l)$, $k \in K$. This yields the following lifted set.

$$
\begin{aligned}
\tilde{Z}^{sv}_l = \ \{(z_{lk} \text{ for } k \in K, x^s_{lt} \text{ for } t \in T,\ q^s_i \text{ for } i \in \Pi(l)) :& \\
\textstyle\sum_{k \in K} z_{lk} = 1 & & \text{(4.34a)} \\
\textstyle\sum_{t \in T_k} V^s_{ltk} = z_{lk}, & \forall k \in K & \text{(4.34b)} \\
\textstyle\sum_{i \in \Pi(l)} Q^{zs}_{ilk} - \sum_{t \in T_k} \tilde{c}^s_{lt} V^s_{ltk} \le 0, & \forall\ k \in K & \text{(4.34c)} \\
Q^{zs}_{ilk} \le \mu^s_i\, z_{lk}, & \forall\ i \in \Pi(l),\ k \in K & \text{(4.34d)} \\
\textstyle\sum_{k \in K} Q^{zs}_{ilk} = q^s_i, & \forall\ i \in \Pi(l) & \text{(4.34e)} \\
V^s_{ltk} = x^s_{lt}, & \forall t \in T_k, k \in K & \text{(4.34f)} \\
V^s \ge 0,\ Q^{zs} \ge 0\}. & & \text{(4.34g)}
\end{aligned}
$$

Proposition 4.3.3. Let Z^s_l and $\tilde{Z}^{sv}_l$ be given by (4.27) and (4.34), respectively, for any $l \in L$ and $s \in S$. Then $\tilde{Z}^{sv}_l \supseteq Z^s_l$, and the extreme points of $\tilde{Z}^{sv}_l$ yield binary values for the z-variables.

Proof: For any given $(\bar{z}, \bar{x}^s, \bar{q}^s) \in Z^s_l$, by letting $\bar{Q}^{zs}_{ilk} = \bar{q}^s_i \bar{z}_{lk}$, $\forall k \in K$, $i \in \Pi(l)$, and $\bar{V}^s_{ltk} = \bar{x}^s_{lt} \bar{z}_{lk}$, $\forall k \in K$, $t \in T_k$, we can easily verify that $(\bar{z}, \bar{x}^s, \bar{V}^s, \bar{q}^s, \bar{Q}^{zs}) \in \tilde{Z}^{sv}_l$. Hence, $\tilde{Z}^{sv}_l \supseteq Z^s_l$. It remains to show that at the extreme points of $\tilde{Z}^{sv}_l$, the z-variables are binary-valued.

Toward this end, consider the linear programming problem to:

$$\text{Minimize } \{c_1 \cdot z + c_2 \cdot x + c_3 \cdot q \ : \ (z, x, q) \in \tilde{Z}_l^{sv}\} \tag{4.35}$$

for any objective gradient vector (c_1, c_2, c_3) for which (4.35) has a unique optimum. To solve (4.35), we can first substitute out for x^s in terms of V^s using (4.34f), and for q^s in terms of Q^{zs} using (4.34e). The objective function then takes the form: $\sum_{k \in K} c_{1k} z_{lk} + \sum_{k \in K} \sum_{t \in T_k} c_{2t} V^s_{ltk} + \sum_{i \in \Pi(l)} (c_{3i} \sum_{k \in K} Q^{zs}_{ilk})$. Now, note that the constraints (4.34b), (4.34c), and (4.34d) are separable in the respective sets of variables $(Q^{zs}_{ilk}, V^s_{ltk})$, $i \in \Pi(l)$, $t \in T_k$, for each $k \in K$, and moreover, each of these separable constraints have their right-hand-sides scaled by z_{lk}. Hence, by LP duality, for any fixed $z \geq 0$ (satisfying (4.34a)), the optimal value of $\sum_{k \in K} \sum_{t \in T_k} c_{2t} V^s_{ltk} + \sum_{i \in \Pi(l)} (c_{3i} \sum_{k \in K} Q^{zs}_{ilk})$ can be obtained as a linear function of $(z_{lk}, \ k \in K)$. Therefore, the linear program (4.35) effectively reduces to:

$$\text{Minimize } \ \{c' \cdot z \ : \ \sum_{k \in K} z_{lk} = 1, \ z \geq 0\} \tag{4.36}$$

for an appropriate objective vector c'. Due to the hypothesis of a unique optimum, and noting the structure of (4.36), the optimal solution $\hat{z}$, say, to (4.36) must be of the form

$$\hat{z}_{lk^*} = 1 \text{ and } \hat{z}_{lk} = 0, \ \forall \ k \in K \setminus \{k^*\}, \text{ for some } k^* \in K, \tag{4.37}$$

which is binary-valued. This completes the proof. □

Similar to the case of Z_l^{sc}, we can construct separation problems based on (4.34) for $\tilde{Z}_l^{sv}$ to generate RLT cuts as per Remark 4.3.1.

We are now in a position to introduce SPFAM2, a model that is in-between SPFAM, which requires x-variables to be binary-valued, and SPFAM1, which relaxes x-variables to be continuous without adding any related tightening constraints. More specifically, we add to SPFAM a set of partial convex hull constraints based on Proposition 4.3.2 corresponding to a set of selected legs $L_Q \subseteq L$ (to be identified later). This yields the following model.

SPFAM2:

$$\text{Minimize} \sum_{s \in S} p^s Q_s(z, \mu^s) \tag{4.38a}$$

$$\text{subject to: (4.1b) and (4.1c),} \tag{4.38b}$$

where, for each $s \in S$,

$$Q_s(z, \mu^s) = \text{Minimum} \quad \sum_{l \in L} \sum_{t \in T} c_{lt} x^s_{lt} - \sum_{i \in \Pi} f^s_i q^s_i + \sum_{t \in T} \psi_t P^s_t \tag{4.38c}$$

$$\text{subject to} \quad (4.1e)\text{-}(4.1g), \tag{4.38d}$$

$$(4.1h) \text{ with } Cap_t \text{ replaced by } \tilde{c}^s_{lt} \text{ using (4.20)}, \ \forall l \in L \setminus L_Q \tag{4.38e}$$

$$(4.1i), \ \forall\, i \in \Pi \setminus \bigcup_{l \in L_Q} \Pi(l) \tag{4.38f}$$

$$(4.28c), \ \forall\, t \in T, l \in L_Q \tag{4.38g}$$

$$(4.28d), \ \forall\, t \in T, \ \forall i \in \Pi(l), \ l \in L_Q \tag{4.38h}$$

$$(4.28e), \ \forall\, i \in \Pi(l), \ l \in L_Q \tag{4.38i}$$

$$(x^s, y^s, q^s, Q^s, P^s) \geq 0. \tag{4.38j}$$

Alternative to using (4.28b)-(4.28f) above for $l \in L_Q$, we can instead use (4.34b)-(4.34g) for $l \in L_Q$ in order to derive a more manageable problem. When solving family-decomposed subproblems, we can derive Benders' cuts similar to (4.23). If constraints (4.28b)-(4.28f) are used for legs in L_Q as defined in (4.38d)-(4.38j), then similar to (4.23) of Proposition 4.3.1, the corresponding Benders cuts take the form:

$$\eta^s \geq \sum_{t \in T} A_t \bar{\phi}^{3s}_t + \sum_{i \in \Pi \setminus \bigcup_{l \in L_Q} \Pi(l)} \mu^s_i (\max_{k \in K^i} \bar{\phi}^{5ks}_i) + \sum_{k \in K} \sum_{l \in L_k} \bar{\phi}^{1s}_l z_{lk}. \tag{4.39}$$

Alternatively, if constraints (4.34b)-(4.34g) are used for legs in L_Q, the corresponding Benders cuts take the form:

$$\begin{aligned} \eta^s \geq & \sum_{t \in T} A_t \bar{\phi}^{3s}_t + \sum_{i \in \Pi \setminus \bigcup_{l \in L_Q} \Pi(l)} \mu^s_i (\max_{k \in K^{iq}} \bar{\phi}^{5ks}_i) + \\ & \sum_{k \in K} (\sum_{l \in L_k \setminus L_Q} \bar{\phi}^{1s}_l z_{lk} + \sum_{l \in L_k \cap L_Q} \bar{\phi}^{Vs}_l z_{lk} + \sum_{i \in \bigcup_{l \in L_Q} \Pi(l)} \mu_i (\max_{k \in K^{iQ}} \bar{\phi}^{Qs}_i) z_{lk}), \end{aligned} \tag{4.40}$$

where, $\bar{\phi}^{Vs}_l$ and $\bar{\phi}^{Qs}_i$ are optimal dual solutions associated with (4.34b) and (4.34d), respectively, and K^{iq} and K^{iQ} are defined similar to K^i in Proposition 4.3.1 as $K^{iq} \equiv \{k \in K \mid i \in \bigcup_{l \in L_k} \Pi(l) \setminus \bigcup_{l \in L_Q} \Pi(l)\}$, $\forall i \in \Pi$, and $K^{iQ} \equiv \{k \in K \mid i \in [\bigcup_{l \in L_k} \Pi(l)] \cap [\bigcup_{l \in L_Q} \Pi(l)]\}$, $\forall i \in \Pi$.

A solution approach for SPFAM2 is proposed in Algorithm 4.3.2 below.

4.3.3 Proposed Solution Approaches

SPFAM1 and SPFAM2 are two continuous variants of SPFAM in Model 4.2.1, having different levels of relaxations. We now state our proposed solution methodologies that apply

Benders' decomposition framework in concert with the above separation and tightening techniques to solve SPFAM1 and SPFAM2 to near-optimality.

Algorithm 4.3.1. (For solving SPFAM1 of (4.19), with the master problem defined in (4.21) and the subproblems SP^s defined in (4.19c)-(4.19g).)

Initialization: Initialize the incumbent solution z^* as null, the incumbent $\nu^* = \infty$, and lower and upper bounds for the continuous LP-relaxation of SPFAM1 as $LB = -\infty$ and $UB = \infty$, respectively. Solve this LP-relaxation of SPFAM1 using Benders' decomposition and, optionally, with a set of initial feasibility cuts (see Remark 4.3.2 below). Specifically, relax z to be continuous in (4.21d), while iterating between the master problem (4.21) and the subproblems SP^s, $\forall s \in S$, until $LB \geq (1-\epsilon)UB$ for some tolerance $\epsilon > 0$ or until a certain maximum number ($\text{cut}^0_{\max}$) of initial Benders' cuts are generated. In this process, whenever a binary-valued z-solution is obtained that yields $\sum_{s\in S} p^s Q_s(z, \mu^s) < \nu^*$, set it as the incumbent z^* and let $\nu^* = \sum_{s\in S} p^s Q_s(z^*, \mu^s)$. Let $(\bar{z}, \bar{x}, \bar{y}, \bar{q})$ be the solution thus obtained along with a set of Benders cuts. If $LB \geq (1-\epsilon)UB$, $\bar{z}$ is binary, and $\bar{x}^s$ are binary and the same for all $s \in S$, stop with the solution as (ϵ-)optimal and prescribe the family and fleet type assignments as given by $\bar{z}$ and $\bar{x}$, respectively. Otherwise, proceed to Step 1.

Step 1: Solve SPFAM1 of (4.19) via Benders' decomposition, using the Benders' cuts obtained above as a set of starting inequalities. In this process, following the implementation of Geoffrion and Graves (1974) and Adams and Sherali (1993), we effectively solve the overall master program using a branch-and-bound algorithm. Whenever a new binary feasible solution for z is found that produces a master program objective value $\sum_{s\in S} p^s \eta^s \leq (1-\epsilon')\nu^*$, for some tolerance $\epsilon' > 0$, we pause and solve the subproblems to generate additional cuts (4.21c), and then continue with the branch-and-bound algorithm for the master problem. In this step, we solve the decomposed subproblems (4.22) for some initial Benders iterations to generate aggregated Benders' cuts (4.23) to be used in (4.21c), until either a certain maximum number ($\text{cut}^{\text{dec}}_{\max}$) of (4.23) are generated, or no more such cuts can be obtained. Subproblems (4.19c)-(4.19g) are then used in the later Benders iterations to obtain Benders' cuts (4.21c). The z-solution used in the (decomposed) subproblems and the weighted sum of the subproblem objective values $\sum_{s\in S} p^s \hat{Q}_s(z, \mu^s)$ are used to update z^* and ν^*, respectively, as necessary, where for each $s \in S$, $\hat{Q}_s(z, \mu^s)$ is calculated from the solutions that evaluate $Q^k_s(z, \mu^s)$ via (4.22), $\forall k \in K$, using $q^s_i \equiv \min_{k\in K^i} q^{sk}_i$, $\forall i \in \Pi$, as discussed in Section 4.3.1. The least lower bound in the branch-and-bound tree of the master problem is used to update LB. We terminate the procedure by proceeding to Step 2 when the relative gap between LB and ν^* is less than some specified $p\%$ of optimality, where $p\% > 100\epsilon'\%$, or if a certain maximum number, say, $\text{cut}^{\text{B}}_{\max}(>\text{cut}^{\text{dec}}_{\max})$, of Benders' cuts have been generated at this step.

Step 2: Prescribe family assignments based on the incumbent z^* solution. Solve Model 4.2.5 using any applicable algorithm (or a mixed-integer program solver) to obtain fleet type assignments. □

Algorithm 4.3.2. (For solving SPFAM2 of (4.38), with the master problem defined in (4.21) and the subproblems SP^s defined in (4.38c)-(4.38j), $\forall s \in S$, with L_Q initialized as $\emptyset$.)

Initialization: Same as the Initialization step in Algorithm 4.3.1, without updating the ν^* value.

Step 1: Augment L_Q with the flight legs l such that the corresponding $\bar{x}^s_{lt}$-values are more fractional as delineated below, and such that the number of paths passing through any of these legs does not exceed a designated number Π_{max} (which partially governs the total number of Q-variables).

$$L_Q \leftarrow L_Q \cup \{l \in L : \sum_{s \in S} (\max_{t \in T} x^s_{lt} - \min_{t \in T} x^s_{lt}) \leq 0.8|S|, \text{ and } |\Pi(l)| \leq \Pi_{max}\}. \tag{4.41}$$

Update the subproblem constraints (4.38d)-(4.38j), $\forall s \in S$, using the current L_Q information. Solve the LP-relaxation of SPFAM2 using the existing set of Benders' cuts to initialize the process. Generate Benders' cuts of type (4.21c) as necessary. Add these new Benders' cuts into the master problem to update the $\bar{z}_{lk}$-values, $\forall l \in L$, $k \in K$, and repeat Step 1 until a certain maximum number ($\text{cut}^{\text{H}}_{\text{max}}$) of Benders' cuts have been generated at this step, or $|L_Q|$ is either not further augmented or exceeds some specified cardinality limit L_{Qmax}.

Step 2: Solve SPFAM2 given by (4.38) as defined based on the set L_Q obtained in the final loop of Step 1 via Benders' decomposition, using the sets of Benders' cuts obtained via the LP solution from the Initialization step and Step 1 as a set of starting inequalities. This process is akin to Step 1 of Algorithm 4.3.1, where, similar to (4.23) in Proposition 4.3.1, we generate a certain maximum number ($\text{cut}^{\text{dec}}_{\text{max}}$) of aggregated Benders cuts (4.39) or (4.40). We terminate the procedure by proceeding to Step 3 when the relative gap between LB and ν^* is less than some specified $p\%$ of optimality, or if a certain maximum number ($\text{cut}^{\text{B}}_{\text{max}}$) of Benders' cuts have been generated at this step.

Step 3: Same as Step 2 of Algorithm 4.3.1. □

There exist many variants of the proposed algorithms. We discuss several of them below.

Remark 4.3.2. *Feasibility Cuts*

The original time-space flight networks are assumed to have the same number of arrival and departure flight arcs for each station to guarantee a circulatory fleet assignment flow that is balanced at each node. However, after the first-stage z-solution is obtained, the

subnetworks are limited to contain only flight legs that are assigned to the pertinent family from the first stage. The resulting subnetworks may well lose the above schedule balance feature for the stations, and thereby render the balance constraints (4.1f) and (4.22c) infeasible. This aspect can be handled via generating feasibility cuts (in addition to optimality cuts) in Benders' decomposition, or equivalently, by using artificial variables in the balance constraints (4.1f) and (4.22c) to assure complete recourse. We can also add some initial cuts in the first stage to help enforce a balanced family-level assignment for each station. For example, let $\mathcal{A}$ denote the set of stations, indexed by a, and let $ba_{la} \equiv \pm 1$, $\forall l \in L$, $a \in \mathcal{A}$ denote flight l beginning/ending at station a, and 0 otherwise. We can then add the following cuts in the first stage:

$$\sum_{l \in L} ba_{la} z_{lk} = 0, \quad \forall a \in \mathcal{A}, k \in K. \tag{4.42}$$

Alternatively, if any given flight subnetwork does not have balanced arrival and departure flights for each station, we can modify the network as follows to assure obtaining the same daily schedule. To this end, we can add deadheading flow arcs from the end of the day at each station to the beginning of the day for every other station for each type-subnetwork, bearing appropriate deadheading costs. Because no passengers are accepted and thus no profit is generated on these flights, flows on these deadheading arcs will be discouraged in the solutions of the subproblems. By generating optimality Benders' cuts as and when needed, the model will therefore try and make family assignments that tend to reduce such deadheading costs as much as possible over the different scenarios. □

Remark 4.3.3. *RLT Cuts*

In Step 1 of Algorithm 4.3.2, instead of directly applying the representation of Z_l^{sc} (or $\tilde{Z}_l^{sv}$), we can solve the corresponding separation problems (4.31) based on Z_l^{sc} (or alternatively, similarly based on $\tilde{Z}_l^{sv}$), and generate cuts

$$\alpha^{xs} x^s + \alpha^{qs} q^s \leq \alpha^s - \alpha^{zs} z \tag{4.43}$$

of the type (4.32) to augment the constraints for each scenario in order to construct SPFAM2. Albeit less tight than directly applying the entire partial convex hull representations, these cuts are more compact in size, and may therefore permit the consideration of enforcing binary solutions for relatively more flight legs. □

4.4 Computational Experiments

Although the demand realizations for all the paths combined can be viewed as a single scenario, there is still a need to conduct a judicious discretization that would yield a good representation of the problem and yet have a manageable number of scenarios. Therefore,

in Section 4.4.1, we first describe a suitable sample scenario generation scheme. Then in Section 4.4.2, we present the results for Algorithms 4.3.1 and 4.3.2 and examine the benefit of the proposed stochastic programming approach over a deterministic approach using some illustrative test cases. Finally, in Section 4.4.3, we demonstrate the effectiveness of the proposed algorithms through a realistically sized problem.

4.4.1 Scenario Generation

Listes and Dekker (2005) also use a two-stage stochastic program to address the benefit of dynamically matching aircraft capacities to passenger demands, although their focus is on deciding the fleet composition. In their scenario generation, they discretize each (leg-level) demand distribution into $|S|$ segments of equal probability, and randomly select a sample demand for each leg from these $|S|$ segments. A combination of all the random samples over the different flight legs composes one particular scenario, and a total of $|S|$ such combinations yields a set of $|S|$ scenarios. Their numerical experiments show that 50 scenarios are sufficient to capture the impact of demand variations, and considering additional scenarios does not lead to any significant changes in the derived solutions.

We use a similar approach to generate scenarios, without explicitly segmenting continuous distributions. To generate a scenario, we randomly sample each path-level demand according to its (assumed truncated normal) distribution. Each composition of these sampled path-level demands constitutes a particular scenario, and we generate $|S|$ such equally likely scenarios.

Remark 4.4.1. A different, and perhaps more representative, scenario generation approach that could be used is through path grouping. For example, suppose that we select some six key cities, e.g., hubs in a hub-and-spoke network, and assign paths that are incident at each such key-city to a single group. Furthermore, assume that the demand on all paths belonging to the same group are positively correlated, taking either some "high" or "low" level of demands simultaneously, and that the demand on paths of different groups are independent, while the remaining paths not belonging to any group always bear the mean demands. Thus, such a demand pattern would yield a total of $2^6 = 64$ equally likely scenarios. □

4.4.2 Computational Results for Some Illustrative Test Cases

To examine the benefit of the proposed stochastic programming approach, we first tested this paradigm on some illustrative cases involving two aircraft families K1 and K2. Family K1 contains two aircraft types T1 and T2, and Family K2 contains three aircraft types T3, T4, and T5. These five types have capacities 120, 158, 141, 120, and 103, respectively, and the number of aircraft of each type are given as 8, 14, 10, 10, and 4, respectively. The subnetwork for each type was constructed to contain 48 flight legs that formed 162 paths,

resulting in 752 variables (including 96 z-variables for family-level assignments and 240 x-variables for type assignments), 614 constraints, and 2,065 nonzeros in a deterministic FAM (for a single scenario). The corresponding stochastic programming formulation SPFAM using 50 scenarios has 33,196 variables (including 96 binary z-variables, and 12,000 x-variables), 18,423 constraints, and 119,788 nonzeros.

Using historic mean path-level data from United Airlines, we generated scenario fare data via a truncated normal distribution having a standard deviation equal to half its mean. Since our test network is a coarser aggregated representation of United's detailed network, we multiplied the mean demand data values extracted by 100 in order to obtain realistic values that make supply and demand comparable. We then generated truncated normal distribution-based demand data having standard deviations equal to 0.5, 1, 2, and 3 times the mean to obtain different test problem instances. We also generated 10, 25, and 50 scenarios using each of these standard deviations, which produced a total of 12 data sets.

During the implementation of the proposed algorithms, the relative gap tolerance between ν^* and LB was set to $p = 1\%$, and Benders' cuts that were generated prior to the most recent 20 iterations were dropped if they were significantly inactive (i.e., produced slack values larger than $\$3,000$) for 40% of the iterations from the time they were first generated. In Algorithm 4.3.1, we set the maximum number of Benders iterations to 20 in the Initialization step ($\text{cut}^0_{\max} = 20|S|$), and we set this limit to 150 in Step 1 ($\text{cut}^{\text{B}}_{\max} = 150|S|$), including 50 Benders iterations for solving family-decomposed subproblems ($\text{cut}^{\text{dec}}_{\max} = 50|S|$) and 150 Benders iterations for non-decomposed subproblems. For Algorithm 4.3.2, we ran the Initialization step for three Benders iterations, and switched to Step 1, which we ran for 17 Benders iterations. We set $L_{Qmax} = 48$ so that there was no actual restriction on the size of L_Q. We also set $\text{cut}^{\text{B}}_{\max}$ in Step 2 to $100|S|$. The termination criterion based on $\text{cut}^{\text{B}}_{\max}$ was reached by only one case because the specified relative gap tolerance (1%) was achieved in all other cases. The relative gap for the case terminated due to reaching $\text{cut}^{\text{B}}_{\max}$ was 1.1%.

The results from applying Algorithm 4.3.1 to SPFAM1 and applying Algorithm 4.3.2 to SPFAM2 were compared in terms of the number of fractional x-variables in the final solutions produced. We also compared the effect of using the representation $\tilde{Z}^{sv}_l$ as defined in (4.34) versus Z^{sc}_l as in (4.28) for formulating SPFAM2.

Table 4.4 summarizes the number of fractional x-variables in the solutions produced using Algorithm 4.3.1, Algorithm 4.3.2 with the representation of $\tilde{Z}^{sv}_l$ in (4.34), and Algorithm 4.3.2 with the representation of Z^{sc}_l in (4.28). When using $\tilde{Z}^{sv}_l$ of (4.34), nearly all the 48 legs were included within the set L_Q at Step 1. The results show a slight improvement over the solution produced by Algorithm 4.3.1 in terms of the number of fractional x-variables. While using Z^{sc}_l of (4.28), the final cardinality of L_Q at Step 1 ranged from 29 to 47, and the number of fractional x-variables was greatly reduced.

To compare the solutions obtained using a deterministic model (DET) versus solving SPFAM via SPFAM2 of (4.38) using Algorithm 4.3.2 with the partial hull constraints of Z^{sc}_l in (4.28), we solved the traditional FAM (see Barnhart et al. (1998)) for the five aircraft

Table 4.4: Number of fractional x-variables using Algorithms 4.3.1 and 4.3.2.

	Total	$\sigma = 0.5\mu$	$\sigma = \mu$	$\sigma = 2\mu$	$\sigma = 3\mu$
Algo 4.3.1	1560	456	394	442	268
Algo 4.3.2 with (4.34)	1557	456	362	386	353
Algo 4.3.2 with (4.28)	599	83	112	122	282

types using mean demand values. We then solved a demand-driven re-fleeting problem DDR (see Sherali et al. (2005)) over the two families for each scenario, using the family assignments prescribed by each of DET and SPFAM2, respectively. The resulting expected profits (negative of the objective values) from these approaches were compared. Table 4.5 lists these profit values and demonstrates the benefit of SPFAM2, which outperformed the deterministic approach in all but one of the 12 data sets. As expected, the advantage of the

Table 4.5: Comparison of expected profits for the deterministic and stochastic approaches.

	$\sigma = 0.5\mu$		$\sigma = \mu$		$\sigma = 2\mu$		$\sigma = 3\mu$	
$\|S\|$	DET	SPFAM2	DET	SPFAM2	DET	SPFAM2	DET	SPFAM2
10	978,988	980,265	952,477	955,429	953,170	965,961	995,767	1,003,535
25	985,770	985,954	1,023,821	1,034,357	1,023,821	1,025,747	1,061,238	1,080,332
50	1,025,286	1,026,391	1,001,148	1,000,860*	1,035,376	1,042,804	1,057,715	1,065,789

*: The stochastic approach outperformed the deterministic approach all but this scenario.

stochastic over the deterministic approach is more evident when the data has larger variance. Note that we restricted the x-variables to be binary-valued in DET because of its relatively smaller size. Hence, despite solving DET closer to optimality as compared with SPFAM2, the explicit consideration of uncertainty by the latter stochastic approach produced more profitable solutions.

Another factor that influences the effectiveness of the stochastic versus the deterministic approach is the relative magnitudes of the capacity and the demand parameters. Table 4.6 lists the profit values obtained when capacity parameters are about 0.75 of those used in Table 4.5, and demonstrates a greater advantage of using the stochastic programming solution over the deterministic solution. Equivalently, when demands are comparatively higher, we can expect enhanced profits from applying the stochastic programming approach.

Table 4.6: Comparison of expected profits for the two approaches using smaller capacity parameters.

	$\sigma = 0.5\mu$		$\sigma = \mu$		$\sigma = 2\mu$		$\sigma = 3\mu$	
$\|S\|$	DET	SPFAM2	DET	SPFAM2	DET	SPFAM2	DET	SPFAM2
10	756,539	760,256	699,589	707,161	672,859	686,142	675,490	693,237
25	762,739	772,195	727,528	741,242	703,383	721,863	718,934	745,872
50	792,020	796,898	730,497	743,051	723,275	739,381	714,786	739,767

4.4.3 Computational Results for a Realistic Large-Scale Test Problem

Recall that for stochastic programming, the so-called deterministic equivalent problem (DEP) of the corresponding stochastic program is the integrated model that contains first-stage constraints and second-stage constraints from all scenarios such that the constraints form a dual-angular structure, as shown in Problem (2.2). To test the effectiveness of the proposed Algorithms 4.3.1 and 4.3.2, we compared them with solving the DEP formulations of SPFAM in Model 4.2.1, and that of SPFAM1 of (4.19) directly using the mixed-integer programming package CPLEX-MIP, Version 8.1. We applied these approaches to a flight network having 900 flight legs, and $1,814$ paths. We again used two families and five aircraft types having the same capacities as specified in Section 4.4.2. The number of aircraft of each type were taken as 40, 75, 54, 58, and 21, respectively. We sampled 50 scenarios using a truncated normal distribution for demands and fares, having means equal to the historic mean of the path-level data obtained from United Airlines, and standard deviations equal to 6 and 0.5 times the means for demand and fare data, respectively.

After node-aggregation (see Hane et al. (1995)), the DEP formulations for the case of 50 scenarios had 508,750 variables (including 1,800 binary z-variables, and 225,000 x-variables), 418,600 constraints, and 1,548,025 nonzeros in the constraints. We implemented the proposed algorithms in AMPL using the CPLEX 8.1. The computations were carried out on a Dell Workstation PWS670 having 2 GB RAM and 3.4 GHz cpu. When solving the DEP formulation for SPFAM and SPFAM1, both these runs were terminated after reaching a 25-hour cpu time limit. At termination, the software was still processing the initial LP relaxation at the root node for each of these runs, and no feasible basis was found.

Using Algorithm 4.3.1 to solve SPFAM1, we reached a 96% optimality level within $17,179$ cpu seconds (< 5 hours) and achieved 98% optimality within $68,647$ cpu seconds (19 hours). Considering that the SPFAM models are solved around three months in advance of departure, this solution time is well acceptable at this early planning stage. In this process, the Initialization step was run for 50 Benders iterations, and Step 1, the branch-and-bound process, was run for 113 Benders iterations, including 10 for the family-decomposed subprob-

lems and 103 for the integrated subproblems. In the final solution, only 3,592 of the 225,000 x-variables were fractional, a ratio of less than 2%. Using Algorithm 4.3.2 to solve SPFAM2, the Initialization step was run for 50 Benders iterations. We set the fractionality coefficient to 0.7 instead of 0.8 and let $\Pi_{max} = 5$ in (4.41), and let Step 1 run for four iterations. The resulting L_Q contained 55 flight legs. The algorithm achieved 95.6% optimality after 21,380 cpu seconds (6 hours), and the final solution contained 2,662 fractional x-variables, a significant reduction despite using the partial convex hull representations for only 55 legs. Model 4.2.5 based on solutions obtained from Algorithm 4.3.1 and Algorithm 4.3.2 were both solved within a second, and reconciled type-level assignment for all scenarios. Furthermore, the expected profit generated from SPFAM1 and SPFAM2 was $\$15,825,114$ and $\$15,846,755$, respectively, while from the solution obtained via the deterministic approach, this figure was $\$15,351,278$. We see a 3% increase in the potential profitability of using SPFAM2, which amounts to about $160 million per year.

4.5 Conclusions and Future Research Directions

We have proposed a two-stage stochastic programming approach for the primary fleet assignment problem that examines different potential demand scenarios so as to prescribe a solution that provides the maximum re-fleeting opportunities for subsequent airline operational stages. The current practice adopted by airlines is to solve the fleet assignment problem using estimated demand data 10-12 weeks in advance. Given the level of uncertainty, deterministic models at this early stage are inadequate to obtain a good match of aircraft capacity with passenger demands. Because the most pertinent information needed from the initial fleet assignment is at the family level, and the type-level assignment is subject to change at the re-fleeting stage according to future demand realizations, our proposed model focuses on assigning aircraft families to the different legs in the flight network while considering aircraft type assignments under different demand scenarios. This model is complemented with a secondary model that performs adjustments within each family, if necessary, to provide a consistent fleet type-assignment information for accompanying decision processes, such as yield management. The primary model is a large-scale 0-1 mixed-integer program, which is computationally prohibitive to solve. Based on a polyhedral study, we proposed two variants for the primary model, each having different levels of relaxation, and we designed suitable solution approaches predicated on Benders' decomposition methodology. Testing on a realistic large-scale problem, the proposed approach took a reasonable solution time of 5 to 19 hours and demonstrated a 3% increase in expected profits over the traditional deterministic approach in use today, which translates to an enhanced annual profitability of $160 million.

We have also proposed in this chapter a robust optimization model that controls decision variation and a model considering recapture, which are more in line with airline operations and yield management strategies, albeit, more complex. For future research, we propose the development of solution approaches for these models so as to be able to handle the

nonlinearity of the objective function for the robust optimization model and the increased size for the recapture model. Also, given the interdependence of the airline fleet assignment with other operations such as schedule planning, aircraft routing, and crew scheduling, more research is necessary on integrating (some of) these features in a single model to assist airlines in their overall decision-making process.

Chapter 5

Two-Stage Workforce Planning Under Demand Fluctuations and Uncertainty

5.1 Introduction

Cost-effective workforce planning has been one of the key issues attracting the attention of management, particularly in labor-intensive service-oriented industries. Two difficulties related to workforce planning are demand fluctuations and demand uncertainty, which are often handled using short-term contractual employees. However, in many cases, because of high training costs, long recruiting and employee learning processes, and management's preference to maintain team consistency, short-term contracts are not the best policies to match the number of employees with workforce demands.

In this chapter, motivated by recent trends in outsourcing and globalization, we address a contemporary workforce planning problem faced by a global financial firm, which seeks to manage multi-category workforce for functional areas located at different service centers (cities), each having office-space and recruitment capacity constraints, and where the workforce demand fluctuates over time and across functional areas due to market uncertainty and dynamic project requirements. The workforce recruitment in the present context targets long-term contractual employees. Management has some limited control over the workforce demand via changing the project implementation centers, months, or shifts, or by splitting demand for certain functional areas so that fractions of the demand are fulfilled at different centers, months, or shifts. To hedge against demand fluctuations and uncertainty, a workforce recruitment and allocation plan is made on a monthly basis for each of the coming months in a rolling horizon fashion until the year-end. For example, right before the beginning of the year, the planning horizon covers the entire twelve months of the coming year;

each month, the plan is rebuilt based on updated information, with the planning horizon extending from the coming month to the end of the year. In order to ensure consistency in the plan from month to month, particularly in regard to personnel decisions, at each planning point, we need to account for the fluctuations and uncertainty in the coming months to the extent possible.

In the problem at hand, we need to make a long-term workforce decision while hedging for the fluctuating and uncertain short-term demand. The problem is formulated as a two-stage stochastic mixed-integer program, in which the first stage makes personnel recruiting and allocation decisions while the second stage, based on the given personnel decision and realized workforce demand, decides on the project implementation assignment. In the remainder of this chapter, we first formulate a deterministic model in Section 5.2. In Section 5.3, we propose a two-stage stochastic programming model and present a suitable solution approach. We report numerical results with insights on applying the deterministic versus the stochastic programming models in Section 5.4. Finally, we close the chapter in Section 5.5 with a summary and some directions for future research.

5.2 Deterministic Model Development

We first consider a deterministic case in which a yearly forecast on the workforce requirement specifies exactly when (months and shifts), where (centers), and how many employees of each category are needed for each function. Two major constraints are the recruitment limit and the facility capacity, due to the labor market size and the number of desks available, respectively. Employees work in shifts, where two adjacent shifts overlap, with the employees for both shifts simultaneously occupying the desk capacity. If the total number of employees needed to work for covering all the functions at a certain time exceeds the desk capacity at a center, or if the labor market for a center in a given month cannot provide the required number of new employees needed of a certain category, then workforce shortage occurs. In this case, tasks of some or all of the functions can be reassigned to be fulfilled by the employees for the *same* corresponding functions in other centers, or during other shifts or months when the number of employees is sufficient. Both forward and backward moves over shifts and months are permissible when tasks are reassigned to be fulfilled by other shifts and months, and moreover, when they are reassigned to other centers, the designated centers must be eligible to perform the corresponding functions. For the type of assignment changes, be it center change, shift change, or month change, managers may have different preference levels. Combinations of center, shift, and month changes in the assignment are also permissible, but undesirable. In our model, we view the workforce demand as the measure of projects or tasks to be fulfilled, and the foregoing subjective preferences are reflected as different weights (penalty costs) assigned to the different types of admissible changes. Hence, we can represent task reassignments using re-directions of workforce demand to other centers, shifts, or months.

The following deterministic model mainly addresses demand fluctuations. In Section 5.3, we extend this to a stochastic model to consider demand uncertainty.

Sets:

F **:** set of functional areas, indexed by f.

I **:** set of centers, indexed by i, with $I(f)$ being the set of centers capable of performing f, $f \in F$.

J **:** set of shifts, indexed by j. For each shift j, we define $J(j)$ as a set that contains shift j and the preceding overlapping shift, except for the first shift of each day, which has $J(j) = \{j\}$.

U **:** set of planning units, indexed by u, with each planning unit $u \equiv (ifj)$ being a triplet of the center, functional area, and shift, where $i \in I(f)$. (We also use $i(u)$, $f(u)$, and $j(u)$ to denote the center, functional area, and shift of the planning unit u, respectively. Unless otherwise specified, we will assume that $u \equiv (ifj)$.)

K **:** set of workforce categories, each having a specific skill set, indexed by k.

T **:** set of planning time units (months), indexed by t, where $T = \{t_1, \ldots, t_n\}$, with t_1 and t_n denoting the first and last months of the planning horizon, respectively. Let $t_0 \equiv t_1 - 1$ denote the time right before the beginning of the first month.

Variables:

x_{ukt} **:** number of employees of category k for planning unit u in month t, after recruitment and relocation, $u \in U$, $k \in K$, $t \in T$; $x_{\cdot\cdot t_0}$ denotes the number of available employees at the beginning of the planning horizon, and is assumed to be a given parameter. (x_{ukt} is also denoted as x_{ifjkt} when center i, functional area f, and shift j need to be specified for the planning unit u; the same notation applies to all variables and parameters.)

h_{ukt} **:** number of new employees of category k, recruited for planning unit u, at the beginning of month t, $u \in U$, $k \in K$, $t \in T$.

$\beta_{ukt}^{i'j't'}$ **:** percentage of workforce demand of category k in planning unit u for month t that is assigned to be satisfied by center i', shift j', in month t', $i' \in I(f(u))$, $j' \in J$, $u \in U$, $k \in K$, $t, t' \in T$.

$q_{ukt}^{i'j't'}$ **:** $\begin{cases} 1, & \text{if a change occurs in the requested workforce of category } k \text{ for unit } u \text{ in month } t, \\ & \text{where } (i'j't') \neq (i(u)j(u)t),\ u \in U,\ i' \in I(f(u)),\ j' \in J,\ k \in K,\ t, t' \in T, \\ 0, & \text{otherwise.} \end{cases}$

p^-_{ukt} **:** employee shortage of category k relative to the required amount, for unit u, in month t, $u \in U$, $k \in K$, $t \in T$.

Parameters:

d_{ukt} **:** demand, i.e., required number of employees of category k for planning unit u, in month t, $u \in U$, $k \in K$, $t \in T$.

sc_{it} **:** office-space capacity for employees of all categories combined at center i in month t, $i \in I$, $t \in T$.

rc_{ikt} **:** recruitment capacity for employees of category k at center i in month t, $k \in K$, $i \in I$, $t \in T$.

α_{uk} **:** expected turnover rate ($\in [0, 1]$) of category k employees, for planning unit u, $u \in U$, $k \in K$.

c^M_{uk} **:** cost of retaining one employee of category k in planning unit u, $u \in U$, $k \in K$.

c^H_{uk} **:** cost of hiring one employee of category k in planning unit u, $u \in U$, $k \in K$.

$c^{\beta i' j' t'}_{ukt}$ **:** penalty cost of re-directing a unit percent of workforce demand for a planning unit u and month t to a different center i', shift j', or month t', or any combination of these three, where $(i'j't') \neq (i(u)j(u)t)$, $u \in U$, $k \in K$, $i' \in I(f)$, $j' \in J$, $t, t' \in T$. (For convenience, we define $c^{\beta ijt}_{ifjkt} \equiv 0$.)

c^q **:** penalty cost on each workforce demand change, regardless of the type of change (being related to shift, month, or center).

c^-_{uk} **:** opportunity cost of having one less employee of category k than required in planning unit u, $u \in U$, $k \in K$.

The deterministic version of the model is presented in Model 5.2.1 below.

Model 5.2.1. Deterministic Workforce Planning Model:

$$\text{Minimize} \quad \sum_{t\in T}\sum_{u\in U}\sum_{k\in K}(c^{M}_{uk}x_{ukt} + c^{H}_{uk}h_{ukt} + c^{-}_{uk}p^{-}_{ukt} + \sum_{\substack{(i'j't')\in(I(f)\times J\times T):\\ (i'j't')\neq(ijt)}} (c^{\beta i'j't'}_{ukt}\beta^{i'j't'}_{ukt} + c^{q}q^{i'j't'}_{ukt})) \tag{5.1a}$$

subject to

$$\textbf{Capacity:} \sum_{\substack{f\in F:\\ i\in I(f)}}\sum_{k\in K}\sum_{j'\in J(j)} x_{ifj'kt} \leq sc_{it}, \qquad \forall i\in I, j\in J, t\in T \tag{5.1b}$$

$$\textbf{Recruitment:} \sum_{\substack{f\in F:\\ i\in I(f)}}\sum_{j\in J} h_{ifjkt} \leq rc_{ikt}, \qquad \forall i\in I, k\in K, t\in T \tag{5.1c}$$

$$\textbf{Transition:} \; x_{ukt} = (1-\alpha_{uk})x_{ukt-1} + h_{ukt}, \quad \forall u\in U, k\in K, t\in T \tag{5.1d}$$

$$\textbf{Shortage:} \sum_{t'\in T}\sum_{i'\in I(f)}\sum_{j'\in J} d_{i'fj'kt'}\beta^{ijt}_{i'fj'kt'} - x_{ifjkt} \leq p^{-}_{ifjkt}, \quad \forall (ifj)\in U, k\in K, t\in T \tag{5.1e}$$

$$\textbf{Percentage:} \sum_{t'\in T}\sum_{i'\in I(f)}\sum_{j'\in J} \beta^{i'j't'}_{ifjkt} = 1, \quad \forall (ifj)\in U, k\in K, t\in T \tag{5.1f}$$

$$\textbf{Change:} \; \beta^{i'j't'}_{ifjkt} \leq q^{i'j't'}_{ifjkt}, \quad \forall (ifj)\in U, (i'j't')\in (I(f)\times J\times T)\setminus\{(ijt)\}, k\in K, t\in T \tag{5.1g}$$

$$(x, h, p^{-}, \beta) \geq 0, \; q \text{ binary}. \tag{5.1h}$$

Interpretation:

Objective: The objective function (5.1a) minimizes the overall total costs for retaining and hiring employees, together with the penalty costs for employee shortages and for workforce demand re-direction. When penalizing changes, we use the sum of $c^{\beta i'j't'}_{ukt}\beta^{i'j't'}_{ukt}$ and $c^{q}q^{i'j't'}_{ukt}$. By adding the term $c^{q}q^{i'j't'}_{ukt}$ on each changed unit, regardless of the change being on shift, month, or center, we can avoid small fractions of changes ($\beta^{i'j't'}_{ukt}$) being unnecessarily spread to many units, and thereby implicitly limit the number of units that incur changes. We could use a more specific cost term $c^{qi'j't'}_{ukt}q^{i'j't'}_{ukt}$ to achieve the same purpose, but as our experience reveals, this results in a significant increase in computational times.

Capacity: The capacity constraints (5.1b) set capacity limits on the total number of employees of all functions, categories, and overlapping shifts combined in each center.

Recruitment: The recruitment capacity constraints (5.1c) set a maximum number of employees of each category that can be hired for each center and month.

Transition: The transition constraints (5.1d) state that the number of employees at any time period equals the number of employees retained from the previous time period plus the number of new-hires.

Shortage: The shortage constraints (5.1e) account for the number of deficit employees. Because it is possible to split the demand for a planning unit and satisfy it via several other planning units, and also to re-direct all or part of the required demand to different months, the demand for unit $u' \equiv (i'fj')$ in month t' satisfied by unit $u \equiv (ifj)$ in month t equals $d_{i'fj'kt'}\beta^{ijt}_{i'fj'kt'}$. We prefer $u' = u$ and $t' = t$, and hence penalize any percentage that is satisfied by other units and months, $\beta^{i'j't'}_{ukt}$, $\forall(i'j't') \neq (ijt)$, in the objective function. The term $\sum_{t'\in T}\sum_{i'\in I(f)}\sum_{j'\in J} d_{i'fj'kt'}\beta^{ijt}_{i'fj'kt'}$ then calculates the total workforce demand for any unit u' and month t' that is covered by unit u in month t. We hope this number to be as close as possible to the number of employees assigned to unit u, namely, x_{ukt}. If no shortage exists, then by minimizing costs related to the number of employees x in the objective function, we have that either the left-hand-side equals 0, that is, the number of employees equals the allocated demand, or, the left-hand-side is negative, in which case, the number of employees carried over from previous periods is greater than demand, and we have supply surplus. The former case is desirable, and in the latter case, no additional penalty other than the salary paid to the extra employees is applied. Else, when demand is greater than the planned number of employees (left-hand-side>0), then workforce shortage exists, and by minimizing the opportunity costs related to p^- in the objective function, the inequality will be satisfied as an equality, and p^- will therefore represent the shortage of employees.

Percentage: The percentage constraints (5.1f) ensure that the entire demand for each planning unit is covered.

Change: The change constraints (5.1g) along with the penalty for the q-variables in the objective function ensure that $q^{i'j't'}_{ukt}$ equals 1 if *any* change occurs to (ukt), and is 0 otherwise. The purpose of the q-variables and the associated objective penalty term is to limit the number of units that incur changes, as discussed above.

Note that the decision variables x and h denote the number of employees, and hence, should be integer variables. To ease computation, we relax them to be continuous in Constraints (5.1h) and round the final solution to the closest integer.

Remark 5.2.1. Model (5.2.1) captures some major practical concerns inherent in the workforce planning problem. It can, however, be extended to further address other considerations. For example, if personnel relocation between centers, functions, and shifts are considered, then we can define a decision variable $y^{u'}_{ukt}$ to represent the number of employees of category k, relocated from planning unit u to unit $u' \neq u$, at the beginning of month t, for $u, u' \in U$,

$k \in K$, $t \in T$. The transition constraints (5.1d) would transform to the following:

$$x_{ukt} = (1 - \alpha_{uk})x_{ukt-1} + h_{ukt} + \sum_{\substack{u' \in U: \\ u' \neq u}} (y^u_{u'kt} - y^{u'}_{ukt}), \quad \forall u, u' \in U, k \in K, t \in T,$$

to reflect the personnel relocation between units. An associated personnel relocation cost should also be added with respect to the y-variables in the objective function.

Sometimes, splitting a project to be fulfilled by multiple centers incurs additional management overhead. In this case, we can define a binary variable $z^{i'}_{ukt}$ such that it equals 1 if the required workforce of category k for unit u in month t is fulfilled by center i', and equals 0 otherwise, for $u \in U$, $i' \in I(f(u))$, $k \in K$, $t \in T$. We can then add a constraint

$$\sum_{t' \in T} \sum_{j' \in J} \beta^{i'j't'}_{ukt} \leq z^{i'}_{ukt}, \quad \forall u \in U, k \in K, t \in T, i' \in I(f)$$

and a penalty term $c^L_{uk}(\sum_{i' \in I(f)} z^{i'}_{ukt} - 1)$ in the objective function, where c^L_{uk} is the penalty cost of splitting workforce demand for a planning unit u of category k to multiple centers, $u \in U$, $k \in K$. As a result, if any request is split to multiple centers, then $\sum_{i' \in I(f)} z^{i'}_{ukt} > 1$, and this splitting will be penalized in the objective function. □

Remark 5.2.2. In our motivating real-life problem, the model is applied to a relatively inexpensive labor market, and the primary concern of the management is less of employee costs and more of obtaining a plan feasible to the capacity constraints with few personnel and assignment changes. Therefore, the employee retention and hiring costs (c^M and c^H) are set to be small constant numbers. Furthermore, the management has expressed preferences regarding the different possible options of changing the project demand assignments. Accordingly, the related penalty costs are designed to match these priorities as follows:

For all $u \equiv (ifj) \in U$, $k \in K$ and $t \in T$, with $i' \neq i$, $j' \neq j$, and $t' \neq t$:

1. Center-change is preferred to shift-change, which is preferred to month-change; hence, we set $c^{\beta i'jt}_{ukt} < c^{\beta ij't}_{ukt} < c^{\beta ijt'}_{ukt}$.

2. Center-and-shift combined changes are preferred to center-and-month combined changes, which are preferred to month-and-shift combined changes; hence, we set $c^{\beta i'j't}_{ukt} < c^{\beta i'jt'}_{ukt} < c^{\beta ij't'}_{ukt}$. Furthermore, we set $c^{\beta ijt'} < c^{\beta i'j't}_{ukt}$ to favor single-type changes over combined changes.

3. The least desirable change is to simultaneously alter the center, shift, and month; hence, we set $c^{\beta ij't'}_{ukt} < c^{\beta i'j't'}_{ukt}$.

4. All types of changes should satisfy workforce demand, and workforce shortage should be avoided if at all possible. Hence, we set $c^{\beta i'j't'}_{ukt} \ll c^-_{uk}$.

In the above, costs for the same types of changes are set to be identical. For instance, all month-only changes have the same penalty costs, regardless of the target units, and all costs for simultaneous month and shift changes are the same and are higher than the month-only change costs. □

5.3 Two-stage Stochastic Programming Model

In reality, demands for future months are not known for certain at the time of workforce planning. We incorporate the demand uncertainty using scenario representations, that is, the realizations of monthly demands in the coming year are assumed to be a set of finite scenarios, denoted as S. Each scenario has a realization probability p^s, for $s \in S$. Accordingly, denote the demand for each scenario s as d^s_{ukt}. Because the recruitment process may last weeks or months, and personnel decisions need to be ascertained well in advance, we need to decide on the number of new-hires for the entire planning horizon "here-and-now", in order to ensure smooth workforce movement. On the other hand, the decision regarding which centers, shifts, or months should be used to fulfill the realized demands is relatively easier to modify and implement. Hence, these decisions, which are unrelated to workforce changes, are second-stage decisions that are subject to demand variations. To reflect the policy corresponding to each realized scenario, we apply the superscript s for the decision variables β, p^-, and q to denote scenario-dependent decisions. The two-stage stochastic program formulation can then be represented as follows.

Model 5.3.1. Two-stage Stochastic Workforce Planning Model:

$$\text{Minimize} \sum_{t\in T}\sum_{u\in U}\sum_{k\in K}(c^M_{uk}x_{ukt} + c^H_{uk}h_{ukt}) + \sum_{s\in S} p^s f_s(x) \tag{5.2a}$$

subject to

$$\sum_{\substack{f\in F:\\ i\in I(f)}}\sum_{k\in K}\sum_{j'\in J(j)} x_{ifj'kt} \le sc_{it}, \qquad \forall i \in I, j \in J, t \in T \tag{5.2b}$$

$$\sum_{\substack{f\in F:\\ i\in I(f)}}\sum_{j\in J} h_{ifjkt} \le rc_{ikt}, \qquad \forall i \in I, k \in K, t \in T \tag{5.2c}$$

$$x_{ukt} = (1-\alpha_{uk})x_{ukt-1} + h_{ukt} \quad \forall u \in U, k \in K, t \in T \tag{5.2d}$$

$$(x, h) \ge 0, \tag{5.2e}$$

where, for each scenario $s \in S$, we have,

$$f_s(x) = \text{minimum} \sum_{t\in T}\sum_{u\in U}\sum_{k\in K}(c^{-}_{uk}p^{s-}_{ukt} + \sum_{\substack{(i'j't')\in(I(f)\times J\times T):\\ (i'j't')\neq(ijt)}}(c^{\beta i'j't'}_{ukt}\beta^{si'j't'}_{ukt} + c^{q}q^{si'j't'}_{ukt})) \tag{5.2f}$$

subject to

$$\sum_{t'\in T}\sum_{i'\in I(f)}\sum_{j'\in J} d^{s}_{i'fj'kt'}\beta^{sijt}_{i'fj'kt'} - p^{s-}_{ifjkt} \leq x_{ifjkt}, \quad \forall (ifj)\in U, k\in K, t\in T \tag{5.2g}$$

$$\sum_{t'\in T}\sum_{i'\in I(f)}\sum_{j'\in J}\beta^{si'j't'}_{ifjkt} = 1, \quad \forall (ifj)\in U, k\in K, t\in T \tag{5.2h}$$

$$\beta^{si'j't'}_{ifjkt} - q^{si'j't'}_{ifjkt} \leq 0, \qquad \forall (ifj)\in U, (i'j't')\in (I(f)\times J\times T)\setminus\{(ijt)\}, k\in K, t\in T \tag{5.2i}$$

$$(p^{s-},\beta^{s}) \geq 0,\ q^{s} \text{ binary}. \tag{5.2j}$$

The first-stage variables are all continuous and implicitly bounded by the capacity. The second-stage problem has binary variables q^s, which pose a challenge when applying Benders' decomposition. However, the purpose of these q-variables is only to enhance the quality of the solution, that is, to limit the number of changes. Although a solution having fractional q-variables is infeasible to the mathematical model, it is still applicable to the workforce planning problem at hand. Hence, we can delete these variables and replace them with a more conservative workforce assignment change policy.

In our motivating problem, assignment changes are prioritized by several distinct levels as described in Remark 5.2.2. At the first level, since only the centers in the set $I(f)$ for function f are eligible to perform function f, and all sets $I(f)$, $\forall f \in F$, contain only a few centers, center-changes will not result in many small fractional assignment changes. Similarly, if a shift-change is necessary to cover a demand request, then only two[1] to three choices are available. Hence, shift-changes will also not lead to many small fractional assignment changes. But month-changes can cause more serious fractionating problems. To this end, we define Δ as the maximum number of months a personnel assignment can be shifted. A workforce demand of month t then can only be covered by the set of months starting from Month $\max\{t-\Delta, t_1\}$ up to Month $\min\{t+\Delta, t_n\}$. Denoting this set of months as $T(t)$, we then limit the candidates for month changes. This will also limit the combined-changes.

Now that both stages contain only continuous variables, we can then directly apply Benders' decomposition. After deciding on the personnel hiring and allocation through the x- and h-variables in the first stage, we can always restore the binary q-variables to make workforce assignment decisions as monthly demand is observed. To apply Benders' decomposition, notice that the second-stage problem (5.2f)-(5.2j) for scenario s is further

[1] A morning shift demand will not be changed to the afternoon shift, its only overlapping shift, since the afternoon shift will not have enough capacity if the demand is spilled from the morning shift. Similarly, the night shift demand cannot be fulfilled by the evening shift.

decomposable for each $f \in F$ and $k \in K$. We can thus solve even smaller sized Benders' subproblem for each $s \in S$, $f \in F$, and $k \in K$ as the following (again with $u \equiv (ifj)$):

$$F_{sfk}(x) = \text{minimum} \sum_{t\in T}\sum_{i\in I(f)}\sum_{j\in J}(c^{-}_{ifjk}p^{s-}_{ifjkt} + \sum_{\substack{(i'j't')\in(I(f)\times J\times T):\\ (i'j't')\neq(ijt)}}(c^{\beta i'j't'}_{ifjkt}\beta^{si'j't'}_{ifjkt})) \tag{5.3a}$$

subject to

$$\sum_{\substack{t'\in T:\\ t\in T(t')}}\sum_{i'\in I(f)}\sum_{j'\in J} d^{s}_{i'fj'kt'}\beta^{sijt}_{i'fj'kt'} - p^{s-}_{ifjkt} \leq x_{ifjkt},\ \forall i \in I(f), j \in J, t \in T \quad \leftarrow \psi_{ifjkt} \tag{5.3b}$$

$$\sum_{t'\in T(t)}\sum_{i'\in I(f)}\sum_{j'\in J}\beta^{si'j't'}_{ifjkt} = 1,\ \forall i \in I(f), j \in J, t \in T \quad \leftarrow \phi_{ifjkt} \tag{5.3c}$$

$$(p^{s-}, \beta^{s}) \geq 0, \tag{5.3d}$$

with the associated dual variables as displayed in (5.3) above. We can generate a Benders' cut from each of these subproblems F_{sfk} as

$$\eta_{sfk} \geq \sum_{i\in I(f)}\sum_{j\in J}\sum_{t\in T}\psi_{ifjkt}x_{ifjkt} + \sum_{i\in I(f)}\sum_{j\in J}\sum_{t\in T}\phi_{ifjkt}, \quad \forall s \in S, f \in F, k \in K. \tag{5.4}$$

The associated cut for the full subproblem is then obtained by simply aggregating (5.4) by summing over indices f and k for each scenario, to get:

$$\eta_s \geq \sum_{ukt}\psi_{ukt}x_{ukt} + \sum_{ukt}\phi_{ukt}, \quad \forall s \in S. \tag{5.5}$$

Incorporating these scenario level cuts (5.5), the Benders' master problem takes the following form:

$$\text{Minimize} \quad \sum_{t\in T}\sum_{u\in U}\sum_{k\in K}(c^{M}_{uk}x_{ukt} + c^{H}_{uk}h_{ukt}) + \sum_{s\in S}p^{s}\eta_s \tag{5.6a}$$

$$\text{subject to} \quad \text{(5.2b)–(5.2e), and}$$

$$\eta_s \geq \sum_{ukt}\psi^{l}_{ukt}x_{ukt} + \sum_{ukt}\phi^{l}_{ukt}, \quad \forall l = 1, \ldots, L_s, \forall s \in S \tag{5.6b}$$

$$\eta_s \geq 0, \quad \forall s \in S : L_s = 0, \tag{5.6c}$$

where the set of L_s Benders' cuts (5.6b) for each scenario s are iteratively generated and indexed by l.

We now propose to solve the two-stage stochastic programming workforce planning model as follows.

Algorithm 5.3.1.

Step 1: Solve the deterministic model 5.2.1, in which we set the demand values for d equal to the expected demand. Let

$$T(t) \equiv \{t' \in T | \max\{t - \Delta, t_1\} \leq t' \leq \min\{t + \Delta, t_n\}\}, \forall t \in T, \tag{5.7a}$$

where, $$\Delta \equiv \max\{|t_1 - t_2| \mid \exists\, \beta_{ukt_2}^{i'j't_1} > 0, \forall t_1 \neq t_2, t_1, t_2 \in T, i' \in I, j' \in J, u \in U, k \in K\}. \tag{5.7b}$$

Step 2: Solve the two-stage stochastic model containing all scenarios using Benders' decomposition, where the Benders' master problem is given by (5.6), and the Benders' subproblems are given by (5.3), for all $s \in S$, $f \in F$, $k \in K$.

Step 3: Based on the x-solution obtained in Step 2, solve the second-stage problems (5.2f)-(5.2j), for all $s \in S$, to derive the β-variable values where the number of changes are limited by the restored binary q-variables. □

5.4 Numerical Experiments

5.4.1 Computational Results for the Deterministic Model

The deterministic model 5.2.1 was applied to a global financial firm that has six overseas service centers. These six centers are located in different cities and have eleven functional areas in total, with each center specializing in one to three of these functional areas. Four shifts are considered: morning, afternoon, evening, and night. Each pair of adjacent shifts overlap for three to four hours, and the employees for both shifts work simultaneously while sharing the same office space. A twelve-month horizon is considered. The turnover rates (the α-parameters) and the workforce shortages (the p^--variables) are negligible and are therefore not considered here. The deterministic model was solved using a Dell Workstation PWS670 having 2 GB RAM and 3.4 GHz cpu. After the above simplification and the pre-solve executed by CPLEX 9.0, the model was reduced to 76,980 constraints and 150,864 variables, including 73,776 binary variables. An ϵ-optimal solution (with $\epsilon = 10\%$) was obtained in 3 minutes. Among the 169 units requesting employees, we obtained 24 center-changes (including 15 request items being split into two centers), eight shift-changes (including four request items being split into two shifts), three month-changes, and three center-and-shift simultaneous changes (all being split instances). The number of changes advocated by the solution amounted to 36% of the requested items. With these changes, 71 more positions could be filled without violating the recruitment capacity, and the better utilized office space resulted in a total saving of 1,045 desks.

5.4.2 Computational Results for the Stochastic Programming Model

To examine the benefit of the proposed stochastic programming approach, we compared it with a deterministic approach. For the former, we sampled various numbers of scenarios using a truncated normal distribution for demands, having means equal to the expected workforce demand data provided by the firm, and standard deviations equal to twice the mean. Using these data, we applied Algorithm 5.3.1 to obtain the number of employees needed to be hired (h-variables) and allocated (x-variables), and the fraction of workforce needed to be reassigned to other units under each workforce demand scenario (β^s-variables). In the deterministic approach, we solved Model 5.2.1 using the expected workforce demand data to obtain the number of employees needed to be hired and allocated, and then fixed the obtained x-values and solved the second-stage problems (5.2f)–(5.2j) for all scenarios $s \in S$, to obtain the fraction of workforce needed to be reassigned to other units under each workforce demand scenario. As a comparison, Table 5.1 lists the sizes of the problems we tested, the average number of changes and splits per scenario made on the workforce requirement, and the average resulting workforce shortage per scenario. When the number of scenarios

Table 5.1: Comparison of assignment alterations and workforce shortage using stochastic programming (SP) and a deterministic approach (DET).

	changes		splits		shortage	
$(\|F\|, \|I\|, \|T\|, \|S\|)$	SP	DET	SP	DET	SP	DET
(11 , 6 , 12 , 15)	151.53	119.40	164.53	130.33	0	0.00
(11 , 6 , 12 , 25)	157.80	140.44	170.36	142.84	0	2.20
(7 , 4 , 6 , 50)	3.78	16.20	5.78	18.80	0	0.00
(11 , 6 , 6 , 50)	52.46	82.74	68.10	74.74	0	3.88
(7 , 4 , 6 , 100)	3.93	9.04	5.93	15.25	0	0.04
(11 , 6 , 6 , 100)	53.96	60.37	69.78	58.27	0	3.49
(7 , 4 , 6 , 150)	3.62	12.38	5.71	16.60	0	0.16
(7 , 4 , 6 , 200)	3.55	8.34	5.82	13.15	0	0.37
(7 , 4 , 6 , 250)	3.87	7.54	6.35	12.33	0	0.14
(7 , 4 , 6 , 300)	3.27	12.32	5.13	15.16	0	0.13

was relatively small (15–25), as in the first two cases, the deterministic approach using expected demand resulted in few changes and splits. This is because the binary q-variables in Model 5.2.1 tend to limit these changes for the expected scenario, and when the number of scenarios is small, the x-solution obtained from Model 5.2.1 is less likely to induce changes over the different scnearios as well. However, when the number of scenarios exceeded 25, the deterministic approach resulted in more changes, splits, and more undesirably, additional workforce shortages. The stochastic programming approach, on the other hand, did not incur a workforce shortage for any scenario.

We also compared the solution times for executing Algorithm 5.3.1, versus using a linear programming package CPLEX 9.0 to solve the deterministic equivalent form of Problem (5.2) (denoted as DEP) that deletes the q-variables and the constraints (5.2i). Whenever the deterministic equivalent form could be solved by CPLEX, the solution time was very short. However, most of the instances were unsolvable due to computer memory limitations, as summarized in Table 5.2. Algorithm 5.3.1, on the other hand, performed more consistently and solved all instances to the specified 90% optimality.

Table 5.2: Solution time comparison for Algorithm 5.3.1 vs. solving DEP via CPLEX 9.0

$(\|F\|, \|I\|, \|T\|)$	11,6,12	11,6,12	7,4,6	11,6,6	7,4,6	11,6,6	7,4,6	7,4,6	7,4,6	7,4,6
$\|S\|$	15	25	50	50	100	100	150	200	250	300
Algo 5.3.1(min.)	628	929	18	256	69	666	136	214	280	420
DEP(sec.)	22	$-^*$	17	$-^*$	36	$-^*$	$-^*$	$-^*$	$-^*$	$-^*$

$-^*$ indicates that the corresponding instance was not solved due to computer memory limitations.

5.5 Summary and Conclusions

In this chapter, we have addressed a multi-category workforce planning problem for functional areas located at different service centers, each having office-space and recruitment capacity constraints, and where the workforce demand fluctuates over time and across functional areas. We began with a deterministic model that deals with workforce fluctuations and in which a yearly forecast on the workforce requirement specifies exact workforce demand. To hedge against demand fluctuations and uncertainty, we proposed a two-stage stochastic program, in which the first stage makes personnel recruiting and allocation decisions while the second stage, based on the given personnel decision and realized workforce demand, reassigns workforce demand among all the units. The second stage of the proposed model contains binary variables that are used to compute and also limit the number of changes to the prescribed plan. Since these variables are concerned with only one quality aspect of the resulting workforce plan and do not affect feasibility issues, we replaced these binary variables with some conservative workforce assignment change policies to obtain more manageable subproblems that contain purely continuous variables.

Computational results were reported for the deterministic case and for the stochastic programming approach. Numerical experiments revealed that when the number of scenarios are nontrivial, the stochastic programming approach results in significantly fewer alterations to the prescribed workforce plan. However, when relatively few scenarios are present, a deterministic approach based on the expected demand is sufficient. We also compared the Benders' decomposition-based solution approach with solving the deterministic equivalent form directly using a commercial linear programming package CPLEX 9.0. Although this

popular software was able to solve the relatively smaller-sized problems much faster, it failed to produce even a feasible solution due to memory limitations when the size of the problem increased. The proposed algorithm, on the other hand, consistently solved all the practical-sized test problems with reasonable effort.

Chapter 6

Research Contributions and Future Research

6.1 Research Contributions

In this research, we have proposed a decomposition-based branch-and-bound algorithm (DBAB) to solve two-stage stochastic programs having mixed-integer first- and second-stage variables. Since most of the existing algorithms for stochastic mixed-integer programs require purely continuous or purely integer variables in at least one of the two stages, and often also assume deterministic technology or recourse matrices, this research is a ground-breaking effort in solving the most generally formulated stochastic mixed-integer programs. The proposed DBAB algorithm adopts a hyperrectangular partitioning process in the projected space of the first-stage variables. Lower bounds for the nodal problems in the branch-and-bound tree are computed by applying a modified Benders' approach that coordinates a master program with lower-bounding scenario-based second-stage subproblems. Each of these subproblems is derived by sequentially constructing a certain partial convex hull representation of the two-stage solution space. We have shown that the convexification (RLT) cuts derived for a given hyperrectangle Ω at any node are reusable in subsequent solutions of the subproblems for each scenario at this node by updating the first-stage variable values. Furthermore, these cuts can be inherited by the subproblems of the children nodes of Ω. Likewise, the Benders' cuts derived for a given Ω can also be inherited by the lower bounding master programs solved for the children nodes of Ω in the enumeration tree for Algorithm DBAB. The overall process is proven to converge to a global optimum for the underlying stochastic mixed-integer problem. We have illustrated the proposed algorithm using a numerical example, and have reported encouraging computational results. The sizes of the deterministic equivalent of our test problems range from having 386 continuous variables, 386 binary variables, and 386 constraints, up to 1795 continuous variables, 1539 binary variables, and 1028 constraints. The results reveal an average savings in computational effort by a factor of 9.5

in comparison with using a commercial mixed-integer programming package (CPLEX 8.1) on a deterministic equivalent formulation.

Next, we studied two contemporary applications of the class of two-stage stochastic mixed-integer programming problems with recourse, namely, the airline fleet assignment problem and the workforce planning problem. In both applications, the stochastic programming approach demonstrated a significant advantage over the deterministic approach, in terms of enhanced profits in the first case and retaining a more stable system in the second case.

In the airline fleet assignment problem, because certain related crew scheduling regulations require early information regarding the type of aircraft serving each flight leg, the current airline practice is to assign aircraft types to the different legs in the flight network using estimated demand data 10-12 weeks in advance. Given the level of uncertainty, deterministic models at this early stage are inadequate to obtain a good match of aircraft capacity with passenger demands. Since the decision that needs to be made "here and now" for crew scheduling purposes is at the family level, and the type-level assignment is subject to change at the re-fleeting stage according to future demand realizations, we accordingly formulated a two-stage SMIP in which the first stage makes only family-assignment decisions, while the second stage performs re-fleeting based on the first-stage decisions and the market demand realizations. This model was complemented with a secondary model that performs adjustments within each family, if necessary, to provide a consistent fleet type-assignment information for accompanying decision processes, such as yield management. The proposed model is a large-scale 0-1 mixed-integer program, which is computationally prohibitive to solve. Based on a polyhedral study, we proposed two variants for the primary model, each having different levels of relaxation, and we designed suitable solution approaches for these formulations predicated on Benders' decomposition methodology. Test results using realistically large-scale problems having up to 900 flight legs and $1,814$ paths, as obtained from *United Airlines*, revealed that the proposed stochastic modeling approach increased daily expected profits by about 3% (which translates to about $160 million per year), in comparison with the traditional deterministic model in present usage that considers only the expected demand. Using this methodology, only 1.6% and 1.2% of the second-stage binary variables turned out to be fractional in the proposed two model variants, respectively. Furthermore, when attempting to solve the deterministic equivalent formulation for these two variants using a commercial mixed-integer programming package (CPLEX 8.1), both the corresponding runs were terminated after reaching a 25-hour cpu time limit, without obtaining a solution for the linear programming relaxation of the model. Using the proposed algorithms, on the other hand, the solution times were significantly reduced to 5 and 19 hours for the two variants, respectively. Considering that the fleet assignment models are solved around three months in advance of departure, this solution time is well acceptable at this early planning stage, and the improved quality in the solution produced by considering the stochasticity in the system is indeed highly desirable.

In the workforce planning problem prompted by a global financial firm, we addressed

multi-category workforce planning for functional areas located at different service centers, each having office-space and recruitment capacity constraints, and facing fluctuating and uncertain workforce demand. To hedge against demand fluctuations and uncertainty, we formulated a two-stage SMIP in which the first stage makes personnel recruiting and allocation decisions while the second stage, based on the given personnel decision and realized workforce demand, decides on the project implementation assignments. The proposed formulation captures the principal practical concerns inherent in the workforce planning problem, and can be easily extended to further address other company-specific considerations. The objective function is flexible enough to reflect different management preference structures, and thus provides a means for conducting sensitivity analyses and exploring various "what-if" scenarios. This two-stage model is again a large-scale stochastic 0-1 mixed-integer program. Binary variables exist in the second stage, and are used to compute and limit the number of changes to the prescribed plan. Since these variables are concerned with only the quality aspect of the resulting workforce plan and do not affect feasibility issues, we replaced these binary variables with certain conservative policies regarding workforce assignment change restrictions in order to obtain more manageable subproblems that contain purely continuous variables. We designed a Benders' decomposition-based algorithm to solve this two-stage stochastic mixed-integer program. Numerical experiments revealed that the stochastic programming approach results in significantly fewer alterations to the prescribed workforce plan when the number of scenarios is nontrivial. When using a commercial linear programming package CPLEX 9.0 to solve the deterministic equivalent form directly, except for a few small-sized problems, this software failed to produce solutions due to computer memory limitations, while the proposed Benders' decomposition-based solution approach consistently solved all the practical-sized test problems with reasonable effort.

6.2 Future Research

For future research in the context of solving two-stage stochastic mixed-integer programs, we recommend investigating extensions of the proposed DBAB algorithm to solve stochastic programs having *general* mixed-integer (in lieu of 0-1 mixed-integer) variables in either or both of the stages, without employing binary transformations. Nonlinear terms, such as risk-measure terms, can be included in the objective function. A further step would be to adapt the proposed algorithm for solving the two-stage robust optimization problem, such as Model 4.2.3 in Section 4.2.1. Implementation of the DBAB algorithm and its extensions to solve real-life large-scale applications, potentially in a parallel computational environment, is another future research direction.

We have proposed a fleet assignment model formulation considering the recapture effect. Algorithmic developments for this formulation and testing for recapture benefits are postponed as future research topics. Similar to the many extensions proposed for the deterministic FAM (see Section 2.2.1), maintenance considerations, flight time-windows, and

varied pricing for different classes of customers can also be included in the stochastic fleet assignment model. Due to the scale of the two-stage stochastic models, however, to implement a near-exact algorithm directly on the entire problem is a formidable venture. A suitable alternative might be to resort to parallel computations, which we also relegate to future consideration.

Finally, regarding the workforce planning problem, we currently define demand in terms of the number of employees that are needed. We can also treat the demand as some measure of targeted service or production levels, and integrate workforce planning with service or production scheduling. As an example, in a call center, we can characterize the percentage of calls handled during each shift as the demand, and thus determine personnel allocation and scheduling decisions simultaneously. This integration of workforce planning and scheduling is an interesting future research direction that is worth exploring.

Bibliography

Abara, J.: 1989, Applying integer linear programming to the fleet assignment problem, *Interfaces* **19**(4), 20–28.

Abernathy, W. J., Nicholas, B., Hershey, J. C. and Wandel, S.: 1973, A three-stage manpower planning and scheduling model – a service-sector example, *Operations Research* **21**(3), 693–711.

Adams, W. P. and Sherali, H. D.: 1993, Mixed-integer bilinear programming problems, *Mathematical Programming* **59**(3), 279–305.

Ahmed, S., Tawarmalani, M. and Sahinidis, N. V.: 2004, A finite branch and bound algorithm for two-stage stochastic integer programs, *Mathematical Programming* **100**(2), 355–377.

Ahuja, R. K., Goodstein, J., Mukherjee, A., Orlin, J. B. and Sharma, D.: 2002, A very large-scale neighborhood search algorithm for the combined through and fleet assignment model, *Technical report*, MIT Sloan School of Management, 4388-01, Cambridge, MA.

Ahuja, R. K., Liu, J., Goodstein, J., Mukherjee, A., Orlin, J. B. and Sharma, D.: 2003, Solving multi-criteria through-fleet assignment models, *in* T. A. Ciriani, G. Fasano, S. Gliozzi and R. Tadei (eds), *Operations Research in Space and Air*, Kluwer Academic Publishers, Dordrecht/Boston/London, pp. 233–256.

Alonso-Ayuso, A., Escudero, L. F. and Ortuño, M. T.: 2003, BFC, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs, *European Journal of Operational Research* **151**, 503–519.

Balas, E.: 1979, Disjunctive programming, *Annals of Discrete Mathematics* **5**, 3–51.

Balas, E., Ceria, S. and Cornuéjols, G.: 1993, A lift-and-project cutting plane algorithm for mixed 0-1 programs, *Mathematical Programming* **58**, 295–324.

Barnhart, C., Boland, N. L., Clarke, L. W., Johnson, E. L., Nemhauser, G. L. and Shenoi, R. G.: 1998, Flight string models for aircraft fleeting and routing, *Transportation Science* **32**, 208–220.

Barnhart, C., Kniker, T. S. and Lohatepanont, M.: 2002, Itinerary-based airline fleet assignment, *Transportation Science* **36**, 199–217.

Bartholomew, D. J.: 1982, *Stochastic Models for Social Processes*, 3rd edn, J. Wiley, Chichester, England.

Bartholomew, D. J. and Forbes, A. F.: 1979, *Statistical Techniques for Manpower Planning*, Wiley, New York, NY.

Beale, E.: 1955, On minimizing a convex function subject to linear inequalities., *the Royal Statistical Society* **17b**, 173–184.

Benders, J. F.: 1962, Partitioning procedures for solving mixed-variables programming problems, *Numerische Mathematik* **4**, 238–252.

Berge, M. E. and Hopperstad, C. A.: 1993, Demand driven dispatch: A method for dynamic aircraft capacity assignment, models and algorithms, *Operations Research* **41**, 153–168.

Birge, J. R. and Louveaux, F.: 1997, *Introduction to Stochastic Programming*, Springer-Verlag, New York, NY.

Birge, J. R. and Louveaux, F. V.: 1988, A multicut algorithm for two-stage stochastic linear programs, *European Journal of Operational Research* **34**(3), 384–392.

Bish, E. K., Suwandechochai, R. and Bish, D. R.: 2004, Strategies for managing the flexible capacity in the airline industry, *Naval Research Logistics* **51**, 654–685.

Blair, C. and Jeroslow, R.: 1978, A converse for disjunctive constraints, *Journal of Optimization Theory and Applications* **25**, 195–206.

Blair, C. and Jeroslow, R.: 1982, The value function of an integer program, *Mathematical Programming* **23**, 237–273.

Carøe, C. C. and Schultz, R.: 1999, Dual decomposition in stochastic integer programming, *Operations Research Letters* **24**(1-2), 37–45.

Carøe, C. C. and Tind, J.: 1997, A cutting-plane approach to mixed 0-1 stochastic integer programs, *European Journal of Operational Research* **101**(2), 306–316.

Carøe, C. C. and Tind, J.: 1998, L-shaped decomposition of two-stage stochastic programs with integer recourse, *Mathematical Programming* **83a**(3), 451–464.

Charnes, A., Cooper, W. W., Lewis, K. A. and Niehaus, R. J.: 1976, A multi-objective model for planning equal employment opportunities, *in* M. Zeleny (ed.), *Multiple Criteria Decision Making, Kyoto, 1975*, 111–134, Springer-Verlag, Berlin, Germany.

Charnes, A., Cooper, W. W., Niehaus, R. J. and Sholtz, D.: 1974, Multi-level models for career management and resource planning, *in* D. J. Clough, C. G. Lewis and A. L. Oliver (eds), *Manpower Planning Models*, 91–112, English University Press, London, England.

Clarke, L. W., Hane, C. A., Johnson, E. L. and Nemhauser, G. L.: 1996, Maintenance and crew considerations in fleet assignment, *Transportation Science* **30**, 249–260.

Corominas, A., Ojeda, J. and Pastor, R.: 2005, Multi-objective allocation of multi-function workers with lower bounded capacity, *Journal of the Operational Research Society* **56**(6), 738–743.

Dantzig, G. B.: 1955, Linear programming under uncertainty, *Management Science* **1**, 197–206.

Dantzig, G. B. and Wolfe, P.: 1960, Decomposition principle of linear programs, *Operations Research* **8**(1), 101–111.

Daskin, M. S. and Panayotopoulos, N. D.: 1989, A lagrangian relaxation approach to assigning aircraft to routes in hub and spoke networks, *Transportation Science* **23**, 91–99.

Desaulniers, G., Desrosiers, J., Dumas, Y., Solomon, M. M. and Soumis, F.: 1997, Daily aircraft routing and scheduling, *Management Science* **43**, 841–855.

Edwards, J. S.: 1983, A survey of manpower-planning models and their application, *Journal of The Operational Research Society* **33**(11), 1031–1040.

Edwards, J. S. and Morgan, R. W.: 1982, Optimal control models in manpower planning, *in* S. G. Tzafestas (ed.), *Optimization and Control of Dynamic Operational Research Models*, 143–175, North-Holland Publishing Company, Amsterdam, The Netherlands.

El Agizy, M.: 1971, A stochastic programming model for manpower planning, *in* D. J. Bartholomew and A. R. Smith (eds), *Manpower and Management Science*, 131–146, The English Universities Press Ltd., London, England.

Erdmann, A., Kiahaschemi, M., Noltemeier, A. and Schrader., R.: 1997, Fleet assignment with respect to itineraries, *International Symposium on Mathematical Programming*, Lausanne, Switzerland.

Farkas, A.: 1995, *The Influence of Network Effects and Yield Management on Airline Fleet Assignment Decisions*, Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Geoffrion, A. M. and Graves, G. W.: 1974, Multicommodity distribution system design by benders decomposition, *Management Science* **20**(5), 822–844.

Georgiou, A. C. and Tsantas, N.: 2002, Modelling recruitment training in mathematical human resource planning, *Applied Stochastic Models in Business and Industry* **18**(1), 53–74.

Glover, F., Glover, R., Lorenzo, J. and McMillan, C.: 1982, The passenger mix problem in the scheduled airlines, *Interfaces* **12**, 73–79.

Gopalan, R. and Talluri, K. T.: 1998, Mathematical models in airline schedule planning: A survey, *Annals of Operations Research* **76**, 155–185.

Graver, J. E.: 1975, On the foundation of linear and integer programming I, *Mathematical Programming* **9**, 207–226.

Grinold, R. C. and Marshall, K. T.: 1977, *Manpower Planning Models*, North-Holland Publishing Company, New York, NY.

Hane, C. A., Barnhart, C., Johnson, E. L., Marsten, R. E., Nemhauser, G. L. and Sigismondi, G.: 1995, The fleet assignment problem: Solving a large-scale integer program, *Mathematical Programming* **70**, 211–232.

Haugland, D. and Wallace, S. W.: 1988, Solving many linear programs that differ only in the righthand side, *European Journal of Operational Research* **37**(3), 318–324.

Hemmecke, R.: 2000, On the positive sum property of Graver test sets. Preprint SM-DU-468, University of Duisburg, http://www.uni-duisburg.de/FB11/disma/ramon/articles/preprint2.ps.

Hemmecke, R. and Schultz, R.: 2003, Decomposition of test sets in stochastic integer programming, *Mathematical Programming, Ser. B* **94**, 323–341.

Higle, J. L. and Sen, S.: 1991, Stochastic decomposition: an algorithm for two-stage linear programs with recourse, *Mathematics of Operations Research* **16**(3), 650–669.

Jacobs, T. L., Smith, B. C. and Johnson, E.: 1999, O&D FAM: Incorporating passenger flows into the fleeting process, *Proceedings of the AGIFORS Symposium*, New Orleans, Louisiana, pp. 128–161.

Jarrah, A. I., Yu, G., Krishnamurthy, N. and Rakshit, A.: 1993, A decision support framework for airline cancelations and delays, *Transportation Science* **27**, 266–280.

Jeroslow, R. G.: 1980, A cutting plane game for facial disjunctive programs, *SIAM Journal of Control and Optimization* **18**, 264–280.

Kall, P. and Wallace, S.: 1994, *Stochastic Programming*, Wiley, Chichester, England.

Klein Haneveld, W. K. and van der Vlerk, M. H.: 1999, Stochastic integer programming: general models and algorithms, *Annals of Operations Research* **85**, 39–57.

Kniker, T. S.: 1998, *Itinerary-Based Airline Fleet Assignment*, Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA.

Kong, N., Schaefer, A. J. and Hunsaker, B.: 2006, Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach, *Mathematical Programming* (to appear).

Laporte, G. and Louveaux, F. V.: 1993, The integer L-shaped method for stochastic integer programs with complete recourse, *Operations Research Letters* **13**(3), 133–142.

Levin, A.: 1971, Scheduling and fleet routing models for transportation systems, *Transportation Science* **5**, 232–255.

Listes, O. and Dekker, R.: 2005, A scenario aggregationcbased approach for determining a robust airline fleet composition for dynamic capacity allocation, *Transportation Science* **39**(3), 367–382.

Martel, A. and Price, W. L.: 1977, A normative model for manpower planning under risk, *in* D. Bryant and R. J. Niehaus (eds), *Manpower Planning and Organizational Design*, 291–305, Plenum, New York, NY.

McClean, S.: 1991, Manpower-planning models and their estimation, *European Journal of Operational Research* **51**(2), 179–187.

Mehlmann, A.: 1980, An approach to optimal recruitment and transition strategies for manpower systems using dynamic-programming, *Journal of the Operational Research Society* **31**(11), 1009–1015.

Morgan, R. W.: 1979, Some models for a hierarchical manpower system, *Journal of the Operational Research Society* **30**(8), 727–736.

Mulvey, J. M., Vanderbei, R. J. and Zenios, S. A.: 1995, Robust optimization of large-scale systems, *Operations Research* **43**(2), 264–281.

Nemhauser, G. L. and Wolsey, L. A.: 1999, *Integer and Combinatorial Optimization*, 2nd edn, Wiley-Interscience, New York /Chichester /Weinheim /Brisbane /Sin- gapore /Toronto.

Ntaimo, L. and Sen, S.: 2005, A branch-and-cut algorithm for two-stage stochastic mixed-integer programs with continuous first-stage variables. Department of Industrial Engineering, Texas A&M University, College Station, TX. Working Paper.

Pilla, V. L., Rosenberger, J. M., Chen, V. C. P. and Smith, B. C.: 2005, A statistical computer experiments approach to airline fleet assignment, *Technical Report COSMOS-05-03*, The University of Texas at Arlington, Arlington, TX 76019 USA.

Price, W. L., Martel, A. and Lewis, K. A.: 1980, A review of mathematical-models in human-resource planning, *OMEGA* **8**(6), 639–645.

Price, W. L. and Piskor, W. G.: 1972, The application of goal programming to manpower planning, *INFOR* **10**, 221–231.

Purkiss, C.: 1981, Corporate manpower-planning - a review of models, *European Journal of Operational Research* **8**(4), 315–323.

Rao, P. P.: 1990, A dynamic-programming approach to determine optimal manpower recruitment policies, *Journal of the Operational Research Society* **41**(10), 983–988.

Rexing, B., Barnhart, C., Kniker, T. S., Jarrah, T. A. and Krishnamurthy, N.: 2000, Airline fleet assignment with time windows, *Transportation Science* **34**, 1–20.

Rockafellar, R. T. and Wets, R. J.: 1974, Continuous versus measurable recourse in N-stage stochastic programming, *Journal of Mathematical Analysis and Applications* **48**, 836–859.

Römisch, W. and Schultz, R.: 2001, Multi-stage stochastic integer programs: An introduction, *in* M. Grötschel, S. O. Krumke and J. Rambau (eds), *Online Optimization of Large Scale Systems*, Springer, Berlin, Germany, pp. 581–600.

Rosenberger, J. M., Schaefer, A. J., Goldsman, D., Johnson, E. L., Kleywegt, A. J. and Nemhauser, G. L.: 2002, A stochastic model of airline operations, *Transportation Science* **36**, 357–377.

Rushmeier, R. A. and Kontogiorgis, S. A.: 1997, Advances in the optimization of airline fleet assignment, *Transportation Science* **31**, 159–169.

Ruszczyński, A.: 1986, A regularized decomposition method for minimizing a sum of polyhedral functions, *Mathematical Programming* **35**, 309–333.

Ruszczyński, A.: 1993, Regularized decomposition of stochastic programs: algorithmic techniques and numerical results. *Working Paper*, Department of Management Science, Rutgers University, Newark, NJ.

Schultz, R.: 1995, On structure and stability in stochastic programs with random technology matrix and complete integer recourse, *Mathematical Programming* **70**(1), 73–89.

Schultz, R.: 2003, Stochastic programming with integer variables, *Mathematical Programming* **97**(1-2), 285–309.

Schultz, R., Stougie, L. and van der Vlerk, M. H.: 1996, Two-stage stochastic integer programming: a survey, *Statistica Neerlandica* **50**(3), 404–416.

Schultz, R., Stougie, L. and van der Vlerk, M. H.: 1998, Solving stochastic programs with integer recourse by enumeration: a framework using Gröbner basis reductions, *Mathematical Programming* **83**(2), 229–252.

Schultz, R. and Tiedemann, S.: 2003, Risk aversion via excess probabilities in stochastic programs with mixed-integer recourse, *SIAM Journal on Optimization* **14**(1), 115–138.

Sen, S.: 2005, Algorithms for stochastic mixed-integer programming models, *in* K. Aardal, G. Nemhauser and R. Weismantel (eds), *Handbooks in Operations Research and Management Science: Discrete Optimization*, Vol. 12, Elsevier, Dordrecht, The Netherlands, chapter 9, pp. 511–558.

Sen, S. and Higle, J. L.: 2005, The C^3 theorem and a D^2 algorithm for large scale stochastic mixed-integer programming: set convexification, *Mathematical Programming* **104**(1), 1–20.

Sen, S. and Sherali, H. D.: 2006, Decomposition with branch-and-cut approaches for two stage stochastic mixed-integer programming, *Mathematical Programming* **106**(2), 203 – 223.

Sherali, H. D.: 1987, A constructive proof of the representation theorem for polyhedral sets based on fundamental definitions, *American Journal of Mathematical and Management Sciences* **7**(3/4), 253–270.

Sherali, H. D. and Adams, W. P.: 1990, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM Journal on Discrete Mathematics* **3**(3), 411–430.

Sherali, H. D. and Adams, W. P.: 1994, A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems, *Discrete Applied Mathematics* **52**(1), 83–106.

Sherali, H. D. and Adams, W. P.: 1999, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*, Kluwer Academic Publishing, Boston, MA.

Sherali, H. D., Bish, E. K. and Zhu, X.: 2005, Polyhedral analysis and algorithms for a demand driven re-fleeting model for aircraft assignment, *Transportation Science* **39**(3), 349–366.

Sherali, H. D., Bish, E. K. and Zhu, X.: 2006, Airline fleet assignment concepts, models, and algorithms, *European Journal of Operational Research* **172**(1), 1–30.

Sherali, H. D. and Fraticelli, B. M. P.: 2002, A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse, *Journal of Global Optimization* **22**, 319–342.

Sherali, H. D. and Shetty, C. M.: 1980, *Optimization with Disjunctive Constraints. Lecture Notes in Economics and Mathematical Systems*, Vol. 181, Springer-Verlag, Berlin, Germany.

Sherali, H. D., Skarpness, B. O. and Kim, B.: 1988, An assumption-free convergence analysis for a perturbation of the scaling algorithm for linear programs, with application to the L_1 estimation problem, *Naval Research Logistics* **35**, 473–492.

Sherali, H. D. and Soyster, A. L.: 1983, Preemptive and nonpreemptive multi-objective programs: Relationships and counter examples, *Journal of Optimization Theory and Applications* **39**(2), 173–186.

Sosnowska, D.: 2000, Optimization of a simplified fleet assignment problem with metaheuristics: Simulated annealing and grasp, *in* P. M. Pardalos (ed.), *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, Kluwer Academic Publishers, Dordrecht, pp. 477–488.

Stougie, L. and van der Vlerk, M. H.: 1997, Stochastic integer programming, *in* M. Dell'Amico, F. Maffioli and S. Martello (eds), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, Chichester, England, pp. 127–141.

Subramanian, R., Scheff, R. P. J., Quillinan, J. D., Wiper, D. S. and Marsten, R. E.: 1994, Coldstart: Fleet assignment at delta airlines, *Interfaces* **24**, 104–120.

Thengvall, B. G., Bard, J. F. and Yu, G.: 2000, Balancing user preferences for aircraft schedule recovery during irregular operations, *IIE Transactions* **32**, 181–193.

Ugwuowo, F. I. and McClean, S. I.: 2000, Modelling heterogeneity in a manpower system: a review, *Applied Stochastic Models in Business and Industry* **16**(12), 99–110.

Van Slyke, R. and Wets, R.: 1969, L-shaped linear programs with applications to control and stochastic programming, *SIAM Journal on Applied Mathematics* **17**, 638–663.

Wagner, H. M. and Whitin, T. M.: 1952, Dynamic version of the economic lot size model, *Management Science* **5**, 89–96.

Walkup, D. and Wets, R.: 1967, Stochastic programs with recourse, *SIAM Journal on Applied Mathematics* **15**, 1299–1314.

Wets, R.: 1966a, Programming under uncertainty: The equivalent convex program, *SIAM Journal on Applied Mathematics* **14**(1), 89–105.

Wets, R.: 1966b, Programming under uncertainty: The solution set, *SIAM Journal on Applied Mathematics* **14**(5), 1143–1151.

Wets, R.: 1988, Large scale linear programming techniques, *Numerical techniques for stochastic optimization*, Vol. 10, Springer, Berlin, Germany, pp. 65–93.

Wollmer, R. M.: 1980, Two-stage linear programming under uncertainty with 0-1 first-stage variables, *Mathematical Programming* **19**, 279–288.

Yadavalli, V. S. S. and Natarajan, R.: 2001, A semi-Markov model of a manpower system, *Stochastic Analysis and Applications* **19**(6), 1077–1086.

Yan, S. and Young, H.: 1996, A decision support framework for multi-fleet routing and multi-stop flight scheduling, *Transportation Research* **30**, 379–398.

Yu, G. and Luo, S.: 1997, On the airline schedule perturbation problem caused by the ground delay program, *Transportation Science* **31**, 298–311.

Yu, G. and Yang, J.: 1998, Optimization applications in the airline industry, *in* D. Z. Du and P. M. Pardalos (eds), *Handbook of Combinatorial Optimization*, Vol. 2, Kluwer Academic Publishers, Boston, pp. 635–726.